ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA
INFORMÁTICA
INGENIERÍA DEL SOFTWARE

SEVICIO DE POSICIONAMIENTO HABITUAL Y RUTINAS DE DESPLAZAMIENTO SEGÚN EL MODELO PEAAS

POSITIONING SERVICES FOLLOWING THE PEAAS MODEL

Realizado por
JOAQUIN CARDOSO BUZÓN
Tutorizado por
CARLOS CANAL VELASCO
Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, SEPTIEMBRE 2016

Fecha defensa:
El Secretario del Tribunal

Resumen:

El principal objetivo de este proyecto es el de poner en práctica el modelo PeaaS (People as a Service), que es un modelo de servicio que tiene como interés principal que el procesamiento de los datos se realicen en los dispositivos móviles de los usuarios, almacenando en ellos un perfil sociológico de su usuario. De este modo se evita que se guarden la información en los servidores y es el propio usuario quien es dueño de su propia información. Evitando la mala circulación de ésta.

A la implementación de este modelo se le va a añadir una serie de funcionalidades que permitirán poner en práctica este modelo y darle un uso cotidiano. Para ello se almacenarán rutinas de desplazamiento del usuario. Estas rutinas, tendrán información de lugares visitados, fecha de la visita y frecuencia entre otras cosas. La aplicación móvil hará uso de estas rutinas para recibir notificaciones push. El sistema aplicará una serie de filtros que serán los encargados de determinar si la notificación es del interés del usuario para mostrarla o rechazarla.

Éste sistema se puede extender a infinitos campos a parte del que ponemos en práctica, como puede ser la detección del tipo de lugar al que visita el usuario (bar, hospital, centro comercial, supermercado, etc) y determinar gustos y aficiones, ampliando el impacto del modelo.

Palabras clave: PeaaS (People as a Service), Smartphone, perfil sociológico, rutinas de desplazamiento.

Abstract:

The main objective of this project is to implement the model PeaaS (People as a Service). Is a service model that has main objective data processing are performed on mobile devices This will prevent information on the servers are saved and the user who owns his own information. Avoiding wrong circulation of this.

The implementation of this model is going to add a number of features that will implement this model and give everyday use They are stored user routines displacement. These routines have information of places visited, date of visit and frequency between other things. The mobile application will use these routines to receive push notifications. The system will apply a series of filters that will be responsible for determining whether the notification is in the interest of the user to display it or reject it.

This system can be extended to infinite fields also we practice, such as detecting the type of place you visit the user (pub, hospital, shopping mall, supermarket, etc.) and determine tastes and interests, expanding the impact of model.

Keywords: PeaaS (People as a Service), Smartphone, sociological profile, scroll routines.

Índice

1 Introducción

Hoy en día, los smartphones han revolucionado la forma de vivir de las personas, pudiendo indicar que son uno de los inventos más importantes de nuestra era. Tal ha sido su impacto, que forman parte fija de nuestra vida diaria, acompañándonos en todo momento y a todas partes. Con los avances producidos en sus aplicaciones, almacenan información de todo lo que hacemos, visitamos, buscamos y toda nuestra interacción con el medio a través de internet. Consecuentemente todas estas acciones permiten que los smartphones sean una parte más de nosotros, convirtiendo lo que almacenan en un auténtico histórico de nuestro día a día. Es por esta causa por la que desarrollamos el siguiente proyecto enfocado a la recopilación en el propio teléfono móvil de la información de su usuario para poder realizar una serie de acciones determinadas que se describirán a lo largo de este documento.

1.1 Motivación

En la actualidad, el procesamiento de la información se lleva a cabo siguiendo el tradicional modelo cliente/servidor. La información generada por los usuarios se almacena y procesa en los servidores de las compañías responsables de las aplicaciones que estos utilizan (Google, Facebook, Instagram, etc.) que son las que construyen y explotan económicamente el perfil sociológico de los mismos. Este modelo plantea varios inconvenientes. En primer lugar, el hecho de que la información está distribuida en diversas plataformas sociales, que solo pueden generar un perfil sociológico parcial y en ocasiones contradictorio de sus usuarios. En segundo lugar, que el usuario no controla quién y con qué objeto procesa su perfil sociológico o accede al mismo, ni obtiene beneficios de su explotación, más allá del uso gratuito de la mayoría de aplicaciones de redes sociales.

Para paliar estos inconvenientes, se han planteado modelos alternativos para el desarrollo de aplicaciones sociales para dispositivos móviles. En concreto, People as a Service (PeaaS) [1], plantea un modelo computacional centrado en el dispositivo móvil que permite que el perfil sociológico del usuario se genere, almacene y proporcione de forma segura como servicio a terceros desde el propio dispositivo. Esto permite a su propietario ser plenamente consciente y capaz de controlar quién y cuándo accede a su información, y proporciona una manera excelente de representar información sociológica colectiva.

En este proyecto se pretender desarrollar un servicio que haga uso de la información que almacenan nuestro Smartphone de nosotros, pudiendo establecer un perfil nuestro y permitir el recibo de información que mayor interés nos pueda entregar, según los datos que tiene nuestros.

[1] J. Guillen, J. Miranda, J. Berrocal, J. Garcia-Alonso, J.M. Murillo, C. Canal. “People as a Service: A Mobile-centric Model for Providing Collective Sociological Profiles”:

1.2 Objetivos

El objetivo del presente TFG es profundizar en los resultados obtenidos previamente, haciendo más preciso el cálculo de lugares de estancia y rutinas de desplazamiento, de forma que pueda ser aplicado a la población general, más allá del caso de estudio de pacientes de Alzheimer, que es el proyecto del que hereda el nuestro, siguiendo el modelo PeaaS. Para ello, se procederá a revisar las estructuras de datos y algoritmos de análisis desarrollados en el prototipo, mejorando el proceso e incorporando resultados recientes en análisis de rutinas de desplazamiento. También será necesario ofrecer el perfil sociológico obtenido como un servicio a terceros a partir de la plataforma de notificaciones push nimBees, para lo que se realizará una extensión de la misma y una serie de filtros para que el sistema pueda decidir, según la información almacenada, qué decisiones tomar, que en este caso consistirá en mostrar o no la notificación. A su vez se implementará la posibilidad de que el usuario pueda manipular la información que guarda su teléfono, permitiendo de este modo que el usuario tenga un acceso total sobre los datos que se procesan sobre él.

Este TFG se enmarca en una línea de investigación y desarrollo que busca mejorar cómo implementar una forma diferente de almacenamiento y tratamiento de la información que se genera de los usuarios, abriendo camino a nuevos escenarios que permitan la evolución a lo que podríamos denominar la PeaaS (People as a Service), convirtiendo el teléfono móvil en un almacén de la información que se procesa sobre su dueño, minimizando la necesidad de intervención del usuario, siendo implementado conjuntamente con un servicio de notificaciones push que puede ser utilizado para que cualquier usuario pueda enviar estas notificaciones para publicitarse o informar sobre algún acontecimiento. El sistema implementado deberá decidir si la notificación puede interesarle o no a su usuario según una serie de parámetros introducidos y el perfil que va creando el teléfono sobre su usuario.

2 Tecnologías usadas:

Modelo PeaaS

El modelo PeaaS (People as a Service) es un nuevo paradigma de computación centrada en los dispositivos móviles que permite generar perfiles sociológicos de sus dueños y proporcionarlos como servicios de forma segura desde los propios dispositivos. Dentro de este paradigma, la Ingeniería del Software guiada por búsqueda proporciona aportaciones relevantes en dos áreas. Por una parte, las nuevas arquitecturas software habilitadas por el paradigma de People as a Service, facilitan el desarrollo de un nuevo tipo de aplicaciones móviles en el que los dispositivos sean usados como agentes de un sistema de inteligencia de enjambre. Por otra parte, la implementación de este paradigma, con las restricciones impuestas por los sistemas operativos móviles actuales, se enfrenta a una serie de limitaciones que pueden ser abordadas aplicando técnicas de Ingeniería del Software Guiada por Búsqueda.

Aprovechando las capacidades computacionales de los dispositivos móviles se les proporciona la tarea de obtener información de sí mismo (en este caso de sus usuarios o propietarios) y utilizarla para generar perfiles sociológicos. Esta información es almacenada en el propio dispositivo y proporcionada como un servicio para otros sistemas.

Plataforma Nimbees

Nimbees es una plataforma de marketing móvil inteligente que aprovecha la información disponible en los dispositivos móviles acerca de nosotros (actividad, ubicación, preferencias…) para modelar un reflejo digital de quiénes somos. Utiliza esta información para, mediante notificaciones push e iBeacons, permitir a las marcas realizar campañas de marketing más adecuadas y personalizadas a cada consumidor, mejorando la segmentación y las métricas para tomar estrategias adecuadas. A diferencia de otras plataformas, NIMBEES sólo almacena la información personal de cada usuario en su móvil, llevando a cabo el modelo PeaaS (People as a Service) descrito en puntos anteriores. Es una plataforma web de marketing móvil para el envío de campañas a través de apps móviles centrada en conocer mejor al usuario[2].

En este proyecto la intención radica en extender una api que sea capaz de comunicarse con esta plataforma, permitiendo en cada instante que las empresas comerciales envíen información a los usuarios según las localizaciones comunes que hayan tenido los usuarios en determinados momentos, como por ejemplo la visita de un centro comercial o cualquier zona de ocio, permitiendo a las empresas de ese lugar, a informar a los usuarios de información que pueda ser relevante para el

Figura 1: logo Nimbees

[2]http://encuentracapital.es/encuentra_capital_III/nimbees/

usuario, ya sean ofertas, promociones o información que sea de interés, permitiendo crear un nuevo y mejorado sistema de publicidad.

Introducción en Android:

- **¿Qué es Android?**

En los últimos años los teléfonos móviles han experimentado una gran evolución, desde los primeros terminales, grandes y pesados, pensados sólo para hablar por teléfono en cualquier parte, a los últimos modelos, con los que el término "medio de comunicación" se queda bastante pequeño, conocidos como Smartphones.

Es así como nace Android. Android es un sistema operativo y una plataforma software, basado en Linux para teléfonos móviles. Además, también usan este sistema operativo (aunque no es muy habitual), tablets, netbooks, reproductores de música e incluso PC's. Android permite programar en un entorno de trabajo (framework) de Java, aplicaciones sobre una máquina virtual Dalvik (una variación de la máquina de Java con compilación en tiempo de ejecución). Además, lo que le diferencia de otros sistemas operativos, es que cualquier persona que sepa programar puede crear nuevas aplicaciones, widgets1, o incluso, modificar el propio sistema operativo, dado que Android es de código libre, por lo que sabiendo programar en lenguaje Java, va a ser muy fácil comenzar a programar en esta plataforma.

- **Historia de Android**

Fue desarrollado por Android Inc., empresa que en 2005 fue comprada por Google, aunque no fue hasta 2008 cuando se popularizó, gracias a la unión al proyecto de Open Handset Alliance, un consorcio formado por 48 empresas de desarrollo hardware, software y telecomunicaciones, que decidieron promocionar el software libre. Pero ha sido Google quien ha publicado la mayor parte del código fuente del sistema operativo, gracias al software Apache, que es una fundación que da soporte a proyectos software de código abierto [3].

Dado que Android está basado en el núcleo de Linux, tiene acceso a sus recursos, pudiendo gestionarlo, gracias a que se encuentra en una capa por encima del Kernel, accediendo así a recursos como los controladores de pantalla, cámara, memoria flash…

[3] Libro Introducción a Android www.tecnologiaUCM.es

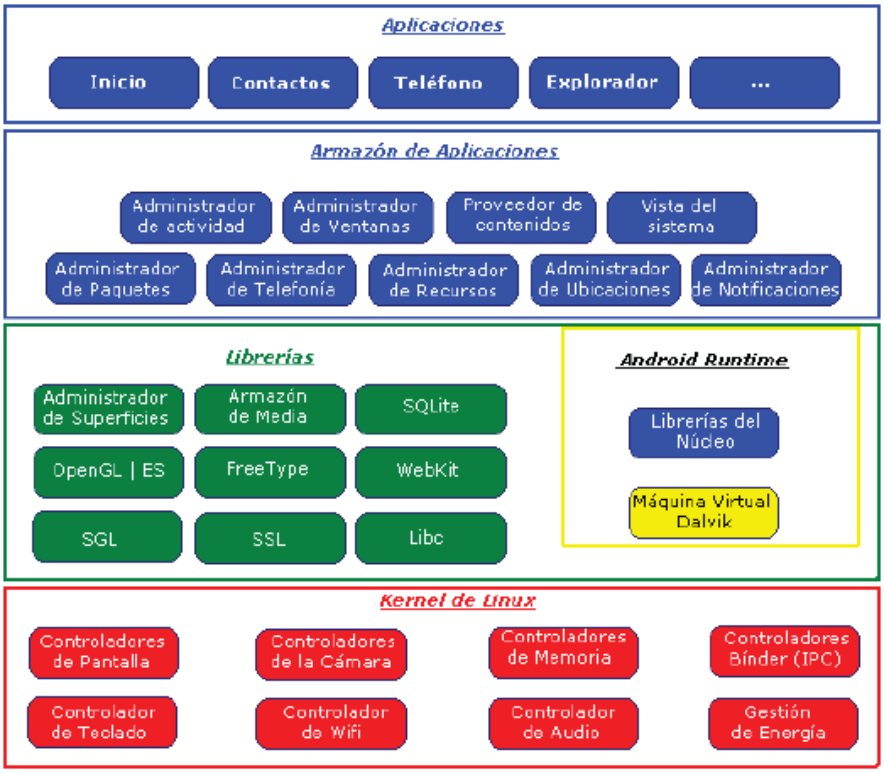

Figura 2: Android

En la imagen se distinguen claramente cada una de las capas: la que forma parte del propio Kernel de Linux, donde Android puede acceder a diferentes controladores, las librerías creadas para el desarrollo de aplicaciones Android, la siguiente capa que organiza los diferentes administradores de recursos, y por último, la capa de las aplicaciones a las que tiene acceso.

- **Servicios**

Los servicios (o service) son tareas no visibles que ejecutan instrucciones en segundo plano. Tienen un hilo propio, lo que permite llevar a cabo cualquier tarea, por pesada que sea. No necesita interfaz, aunque se le puede proporcionar una, en cuyo caso la clase Service la exportaría.

Generalmente se dividen en dos grupos:

-Started - una vez iniciado el servicio se podrá ejecutar de forma indefinida realizando una sola operación cada vez. Se inicia llamando a "startService()".

-Bound - un servicio con el que podremos interactuar enviándole solicitudes, obteniendo resultados e incluso realizar comunicaciones IPC. Se inicia con "bindService()".

Para el desarrollo de nuestro proyecto utilizaremos “Bound” ya que nuestro servicio interactuará a intervalos de tiempo concretos con el sistema, ya sea haciendo uso del

gps para obtener datos de coordenadas y reconociendo los lugares en los que nos encontramos a través de la lógica que implementaremos.

- **Ciclo de vida de un servicio:**

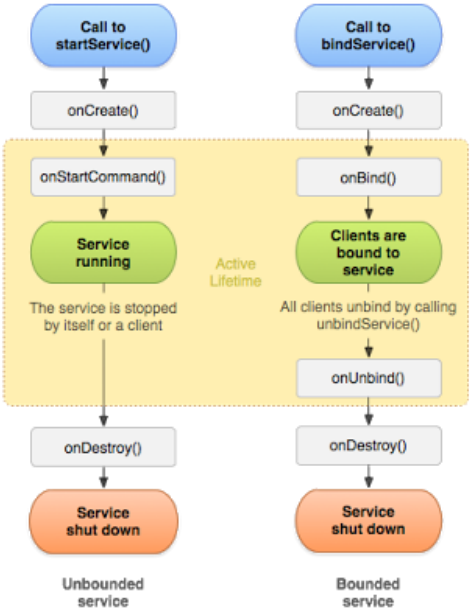

Figura 3: Servicio android

Hay un método común para los dos tipos de servicios que es onCreate(). Normalmente configuraremos aquí nuestro servicio.

Los servicios "Started" se inician de forma indefinida en el método onStartCommand(). Para detener el servicio deberemos llamar a "stopSelf()" desde el mismo servicio o a "stopService()" desde la activity o componente que lo puso en marcha.

En los servicios "Bound" se inicia su vida en el método "onBind()" que sera el encargado de enlazar el servicio con la activity o componente que lo llamo. Se ejecutara de forma indefinida hasta que llamemos a "unbindService()" desde la activity o componente que lo puso en marcha, de esta manera se cierra la conexión y se terminar el servicio, en este momento el servicio pasara al método "onUnbind()".

Otro método común para los dos servicios es "onDestroy()". Sera el lugar donde guardaremos los datos importantes y limpiaremos el servicio de los recursos que hayamos podido utilizar [4].

- **GPS**

Dentro de las diversas herramientas que incluye el sistema operativo Android, para la realización de nuestro proyecto vamos a hacer uso del gps, que nos proporcionará la información necesaria para determinar cuáles son los puntos geográficos que visita el usuario.

[4] http://elbauldeandroid.blogspot.com.es/2013/12/services.html

Para el uso de éste sensor utilizaremos la api que ofrece google “Google maps Apis”, que nos ayudará a determinar los reconocimientos de los recorridos que ejecute el usuario.

- **JExcelApi**

Es una api, escrita en java, que permite leer y escribir hojas de cálculo [5].

Con esta api, se permitirá en la aplicación exportar los datos obtenidos en las rutinas a un archivo Excel o importarlos nuevamente del archivo Excel a la base de datos.

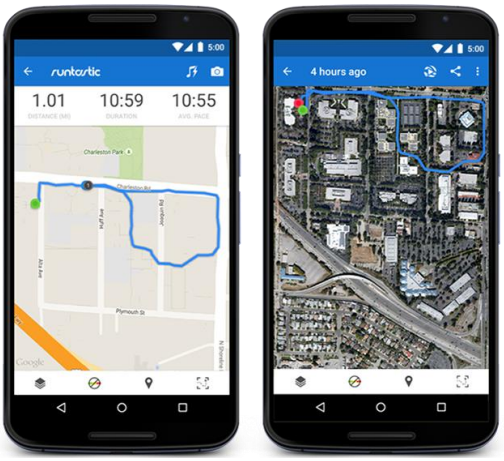

Figura 4: GPS

[5] http://jexcelapi.sourceforge.net/

3-Desarrollo

3.1-Proyecto

Se pretende presentar un sistema basado en el modelo PeaaS (People as a Service), centrado en los dispositivos móviles, que permite generar perfiles sociológicos de sus dueños y proporcionarlos como servicios de forma segura desde sus propios dispositivos. Éste modelo, le ofrece al usuario la posibilidad de almacenar los datos que su dispositivo va generando sobre posicionamiento, búsquedas, etc. evitando que dicha información pase a formar parte de empresas que comercialicen sus datos e información de un lado a otro llegando en ocasiones a hacer un uso fraudulento de ellos.

Los perfiles que se van generando se utilizarán para mostrar información relevante para el propio usuario en forma de notificaciones push.

El escenario elegido para la implantación del modelo PeaaS ha consistido en el desarrollo de dos aplicaciones móviles, una para el envío de mensajes y otra para su recepción. La aplicación destinada al envío le ofrece al usuario una serie de parámetros que este puede ir introduciendo, como pueden ser dirección, mes, día, etc. Estos parámetros se utilizarán en la aplicación de recepción, que según los datos del usuario almacenados en el dispositivo (perfil), filtrará los parámetros, para saber si es relevante para el propio usuario o no.

Con el funcionamiento de la aplicación se pretende dar la posibilidad de que cualquier usuario pueda promocionarse o enviar su propia publicidad teniendo en cuenta parámetros de posicionamiento como pueden ser la latitud y longitud de su establecimiento, llegando la información a los usuarios que viven cerca del lugar o a usuarios que dentro de alguno de sus recorridos diarios pasa por ese sitio o por las personas que frecuentan ir a su establecimiento, etc.

Para el funcionamiento de todo el sistema se ha hecho uso de la api de Nimbees. Dicha api recibe una petición post con unos datos de entrada, entre ellos el mensaje de la notificación e información adicional (filtros) y se envía a los dispositivos seleccionados una notificación push con la información pertinente.

3.2- Descripción de las aplicaciones

-Aplicación 1 (Mis rutinas): se encarga de ir almacenando los lugares frecuentes que el usuario va visitando a lo largo del día (su casa, trabajo, supermercado, tiendas, etc). La aplicación va almacenando la información tanto de los puntos finales como del recorrido que se realiza hasta llegar. Una vez va detectando los lugares frecuentes se va introduciendo la frecuencia con la que se va a esos lugares para poco a poco ir generando el perfil del usuario. La aplicación permite adicionalmente importar la información que la aplicación va almacenando del usuario tanto en un archivo .txt como en un Excel. Además muestra una lista de las últimas notificaciones recibidas. La parte más importante de esta aplicación es la aplicación de filtros sobre las notificaciones recibidas, para evaluar de una forma inteligente si la notificación ha de

ser mostrada al usuario siguiendo los patrones establecidos en el perfil del usuario creado en el dispositivo.

- Aplicación 2 (Envío de Notificación): se encarga de enviar los datos que el usuario emisor dictamine relevantes a la api de Nimbees, por los que la aplicación del usuario receptor filtrará la notificación en base a su información almacenada. Ésta aplicación ha sido desarrollada en forma de apoyo para la demostración del correcto funcionamiento del sistema.

3.3- Actores

Habrá dos actores además del sistema que tendrán acceso a éste.

El usuario principal, que será sobre quien el sistema trabaja y obtiene las rutinas (aplicación "Mis rutinas"), pero recordemos que la idea es tratar de reducir al máximo la interacción de este con el sistema, tratando que la introducción de datos sea mínima o nula, es el sistema automáticamente quien deberá aprender del usuario sin que este se dé cuenta si quiera. Sin embargo, si se le permitirá en cualquier caso hacer algunas consultas en el móvil, como pueden ser la exportación en un archivo Excel o .txt de sus principales rutinas, la visualización de las últimas notificaciones e incluso la manipulación de los datos que se han almacenado sobre él.

El segundo actor será el que se encargue de hacer uso de la otra aplicación (aplicación "Envío de Notificaciones"), para poder enviar las notificaciones a los usuarios finales y probar el correcto funcionamiento del sistema.

El sistema se encargará de ir recopilando las rutinas de desplazamiento del usuario, para posteriormente ir reconociendo los puntos interesantes como pueden ser recorridos y lugares habituales. Además también tiene la tarea de una vez reciba una notificación enviada por un usuario, filtrar según los parámetros recibidos si es de interés para el usuario final o por el contrario desestima mostrarla.

3.4 Requisitos

A continuación se citan los requisitos funcionales y no funcionales a ser tenidos en cuenta en la producción de la aplicación con el fin de intentar definir de forma más concreta el sistema con los requerimientos mínimos que este debería cumplir.

3.4.1 Requisitos Funcionales

- El sistema detectará ante desvíos en la rutina de desplazamiento del usuario, los puntos finales, para reconocer si estos son nuevos o simplemente son desvíos.

- El sistema reconocerá los lugares de destino a través de un radio de localización.

- El sistema permitirá que los datos sean guardados en diferentes formatos.

- El sistema permitirá la recepción de notificaciones push.

- El sistema permitirá el envío de notificaciones push.

- El sistema filtrará las notificaciones push según los criterios de filtrado establecidos.

- El sistema será capaz de obtener el email del usuario de su teléfono para poder realizar el registro en la aplicación.

- El usuario podrá consultar sus rutinas. Para ello deberá acceder a la memoria del teléfono y dentro encontrará documentos con los datos almacenados.

- El usuario podrá manipular los datos de sus rutinas desde el archivo Excel exportado desde la aplicación.

- El usuario podrá importar los cambios realizados en su archivo Excel.

- El usuario podrá recibir notificaciones a través del sistema.

- El usuario podrá enviar notificaciones a través del sistema.

- El usuario podrá consultar las últimas notificaciones recibidas.

- El usuario podrá realizar una exportación de su base de datos.

- El usuario podrá realizar una importación de su base de datos para recuperarla en caso de error.

- El sistema realizará una copia de la base de datos antes de su manipulación para recuperarse en caso de error en el procedimiento.

3.4.2 Requisitos No Funcionales

- El sistema debe estar desarrollado en plataforma Android. Es la plataforma que se ha elegido para el desarrollo en un principio, debido a que se trata de la plataforma más asequible para los usuarios, con el objetivo de que cualquiera pueda tener acceso al servicio sin realizar una gran inversión monetaria.

- El sistema debe cumplir con la LOPD. Es obligatorio cumplir con la Ley Orgánica de Protección de Datos ya que trabajamos con datos sensibles de los usuarios.

- El sistema se adaptará a los recursos disponibles.

- El sistema debe ser lo más eficiente posible optimizando la utilización de los recursos del dispositivo móvil.

3.5 Funcionalidad y Arquitectura

Como se ha descrito, el conjunto de aplicaciones desarrolladas ha heredado parte de funcionalidad un proyecto de fin de grado anterior. Esta funcionalidad ha sido adaptada a la nueva lógica de funcionamiento que debe de presentar el presente escrito. Se ha de tener en cuenta que la app Caremee estaba destinada a enfermos de Alceimer, siendo su principal objetivo detectar recorridos de los enfermos para avisar a su cuidador en caso de que el paciente salga de un recorrido habitual. Describimos la funcionalidad y los cambios aplicados sobre ella:

- Acumulación de datos: dicha funcionalidad se ha adaptado a los requisitos establecidos, ya que estaba enfocada a la detección de rutas de usuarios enfermos de alcéimer. Sobre ella se han realizado el cambio de la api que se utiliza para la detección del posicionamiento que provocaba desvíos significativos en algunas ocasiones, consiguiendo reducir estos desvíos y la mejora de la precisión.

- Monitorización de las rutas: consiste en la detección, según los datos que se van recogiendo de la ruta seguida, de si la ruta realizada es una ruta conocida o no y en caso de no serlo se envía al cuidador un mensaje avisándolo de ello por si el paciente se ha perdido. En nuestro caso se ha eliminado gran parte de esta funcionalidad y sólo se ha tenido en cuenta la detección de la ruta para estudiar si es una ruta conocida o nueva, aumentando la holgura de desvío para no considerar que el usuario ha cambiado de ruta si éste desvía su recorrido por otro lugar para llegar al mismo destino.

- Analizador de datos: Esta tarea convierte lo que en un principio son muestras de latitud, longitud y fecha, en rutinas con una frecuencia, un lugar de partida y de llegada, y una convicción de fiabilidad de la misma. El proceso seguido para dicho análisis o aprendizaje, es el descrito por Davenport y Prusak, basado en la pirámide de conocimiento. A esta funcionalidad se le ha añadido un sistema de recuperación ante fallos que realiza una copia de seguridad de la base de datos para no perderlos en caso de error.

También se ha heredado parte de la base de datos, concretamente la parte encargada del almacenamiento de las rutinas, teniendo en cuenta las modificaciones necesarias para cumplir con los requisitos establecidos.

Notificaciones Push

En términos de tecnología de movilidad, las Notificaciones Push son aquellos mensajes que recibimos en el dispositivo y que han sido emitidos desde cualquier punto de un sistema. Tenemos un ejemplo muy claro en el popular WhatsApp, donde son los usuarios los que envían mensajes a los dispositivos de otros usuarios. Otro ejemplo con notificaciones enviadas de manera automática (y no manual como WhatsApp) podría ser una aplicación de cliente de correo, cuando el servidor detecta un mensaje entrante envía una notificación al dispositivo del usuario.

Las Notificaciones Push permiten el envío de mensajes desde cualquier parte de un sistema a una aplicación móvil tanto si la aplicación está siendo utilizada por el usuario, si está corriendo en un segundo plano, si todavía no ha sido arrancada o, incluso, si el dispositivo está en reposo.

Google Cloud Messages

Las notificaciones Push en Android mediante Google Cloud Messaging (GCM) se ejecutan en un escenario que está compuesto por, al menos, tres actores:

Google Cloud Messaging: El servicio de Google habilitado para el envío de Notificaciones Push a dispositivos Android.

Servidor: con un servicio (REST, SOAP, aplicación web, etc…) que será el encargado de gestionar los identificadores de registro de dispositivos a los que podemos enviar las notificaciones y de comunicarse con GCM solicitando el envío de notificaciones al dispositivo (o dispositivos) deseado. En nuestro caso utilizaremos la api de Nimbees, sobre la que hablaremos más adelante

Dispositivo Android receptor: que recibirá las notificaciones.

Dispositivo Android emisor: que enviará las notificaciones, junto con información adicional, que será procesada en el otro dispositivo.

Para poder enviar notificaciones a un dispositivo Android desde GCM dicho dispositivo debe estar antes registrado en el servicio ofrecido por Nimbees. Dicho paso se realiza de forma transparente al usuario al iniciar éste el sistema. Para ello se ha implementado una función que obtiene la cuenta de correo electrónico de usuario de Android y mediante una petición post realiza el registro con la aplicación implementada en la plataforma Nimbees.

Registro de la aplicación móvil

Para el registro de nuestra aplicación Android, lo primero que debemos hacer para que la aplicación pueda recibir Notificaciones Push desde Google Cloud Messaging (GCM) será registrarla en dicho servicio. Es una forma de decirle a GCM: “soy un dispositivo que quiere recibir notificaciones de una aplicación”. La forma que tenemos de decirle la aplicación de la que queremos recibir notificaciones es indicándole un número de proyecto (lo veremos en el siguiente apartado).

Si todo está correcto, GCM nos responderá con un identificador de registro.

Figura 6: Registro

Para terminar el proceso de registro de la aplicación que hemos creado en la web de Nimbees, introduciremos el identificador obtenido en la plataforma de GCM y ya se podrá comunicar con GCM para indicarle que debe enviar una notificación.

El envío de la notificación.

Podemos enviar una Notificación Push a cualquiera de los dispositivos Android que tengamos registrados en nuestro servicio desde cualquier parte del sistema.

Para ello, nuestro servicio tendrá una operación donde recibirá la información relativa al mensaje que queremos enviar en forma de notificación y el destinatario o destinatarios. La información relativa al destinatario puede ser directamente el identificador del registro u otra información que el servicio sepa relacionar con dicho identificador de registro.

Figura 7: Envío de la notificación

A continuación, con la información de la petición de envío de notificación que recibió nuestro servicio, éste envía una nueva petición a GCM (puede hacerse de manera síncrona o asíncrona: HTTP o XMPP) para que mande la notificación al dispositivo, en nuestro caso se utilizará http. El destinatario de la notificación se indica mediante el identificador de registro que obtuvimos en el punto anterior. Necesitaremos adjuntar unas credenciales de servidor a nuestra petición.

Una vez que hemos enviado la petición a Google Cloud Messaging con nuestras credenciales del servidor, la información relativa a la notificación y su destinatario, GCM enviará dicha Notificación Push.

Habilitando el servicio Google Cloud Messaging.

Para habilitar el servicio Google Cloud Messaging debemos seguir una serie de sencillos pasos:

- **Creando un proyecto Google API.**

Lo primero que haremos será crear un proyecto en Google API, para ello accedemos a la consola de desarrolladores de Google con nuestro usuario.

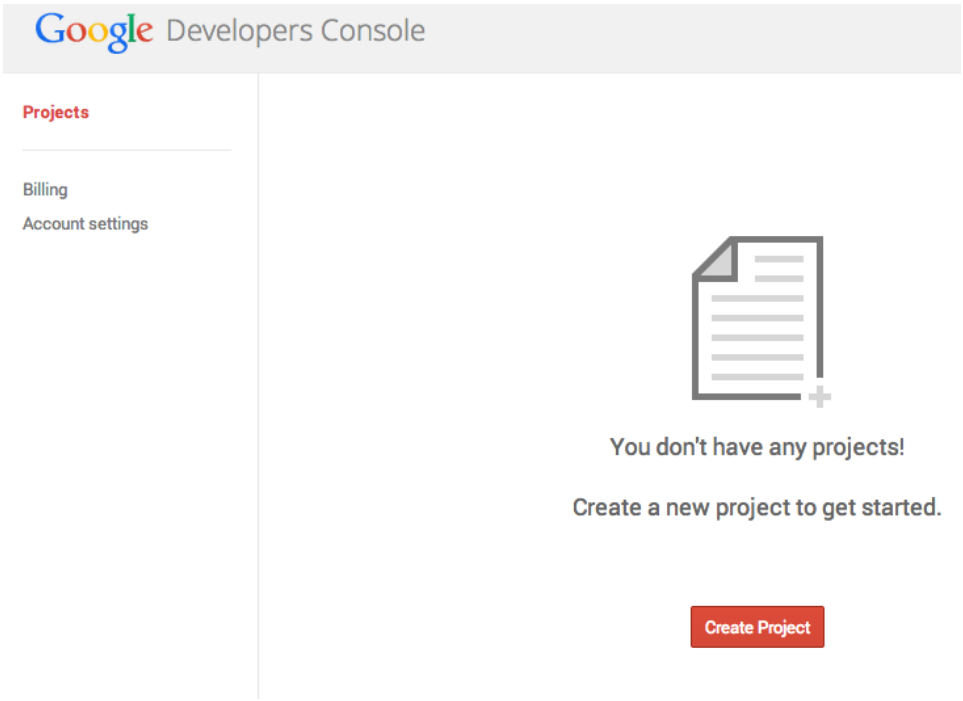

Figura 8: Google cloud Messages

En la sección “Projects” pulsamos sobre el botón “Create Project”, le asignamos un nombre y ya tendremos nuestro proyecto creado. Nos aparecerá una pantalla donde podremos ver información relevante a dicho proyecto.

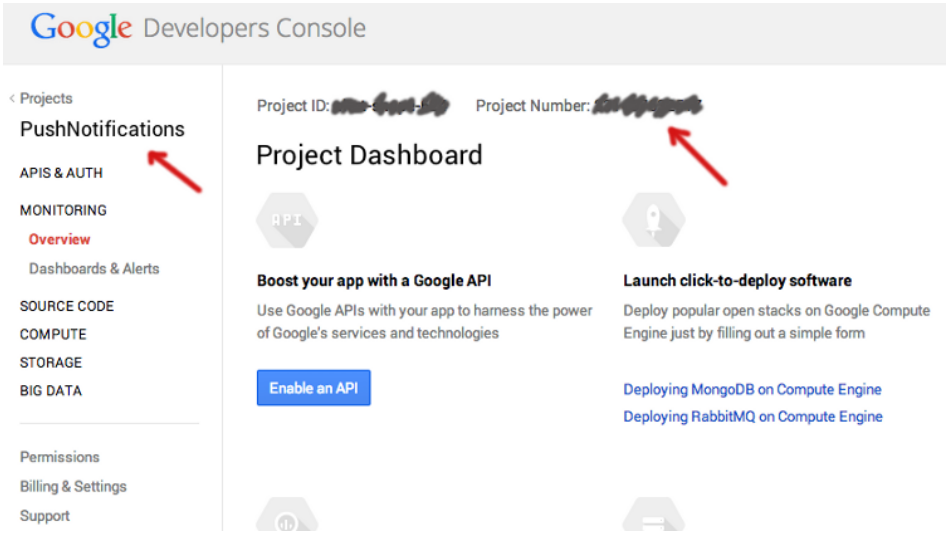

Figura 9: Obtención de project Number

Para las Notificaciones Push, el campo que más nos interesa será Project Number o sender id. Dicho número es necesario para el paso de registro del dispositivo en GCM. Dicho Project Number será algo del tipo: 231447612989

- **Habilitando la opción Google Cloud Messaging.**

Una vez tenemos el proyecto creado, lo siguiente que haremos será habilitar el servicio de Google Cloud Messaging. Para ello, pulsamos en el menú de la izquierda sobre APIS & AUTH > APIs y activamos el servicio "Google Cloud Messaging for Android" (por defecto estará desactivo).

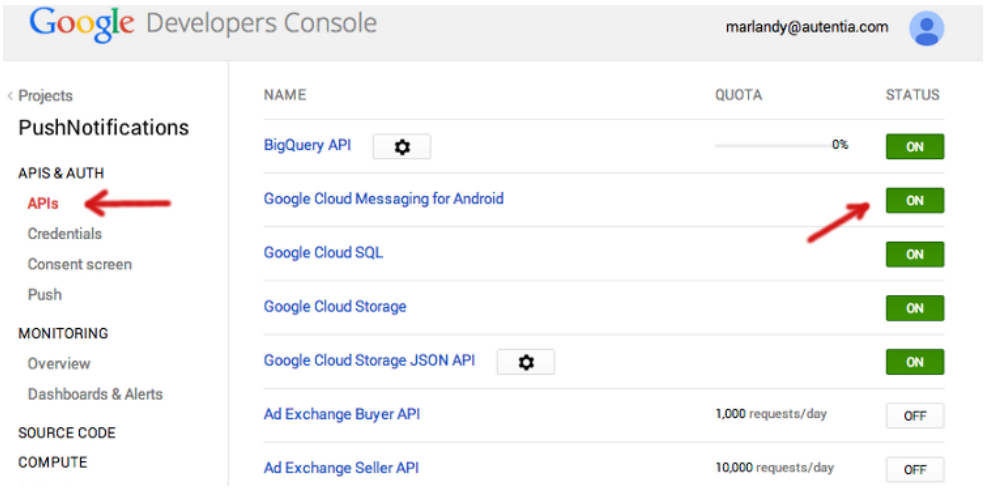

Figura 10: Activar google cloud menssaging

- **Obteniendo la clave de acceso al API.**

Por último, necesitaremos tener una clave de acceso a nuestro API para que podamos enviar peticiones a GCM y que éste las sirva las notificaciones al dispositivo o

dispositivos destino. Esta clave de acceso será utilizada por nuestro servicio tal y cómo vimos en el paso 2 del envío de la notificación.

Para generar la clave de acceso que usará nuestro servidor pulsamos en APIS & AUTH > Credentials y en la sección Public API Access, pulsamos sobre el botón “Create New Key”. En el diálogo que nos aparecerá, pulsamos en “Server Key”.

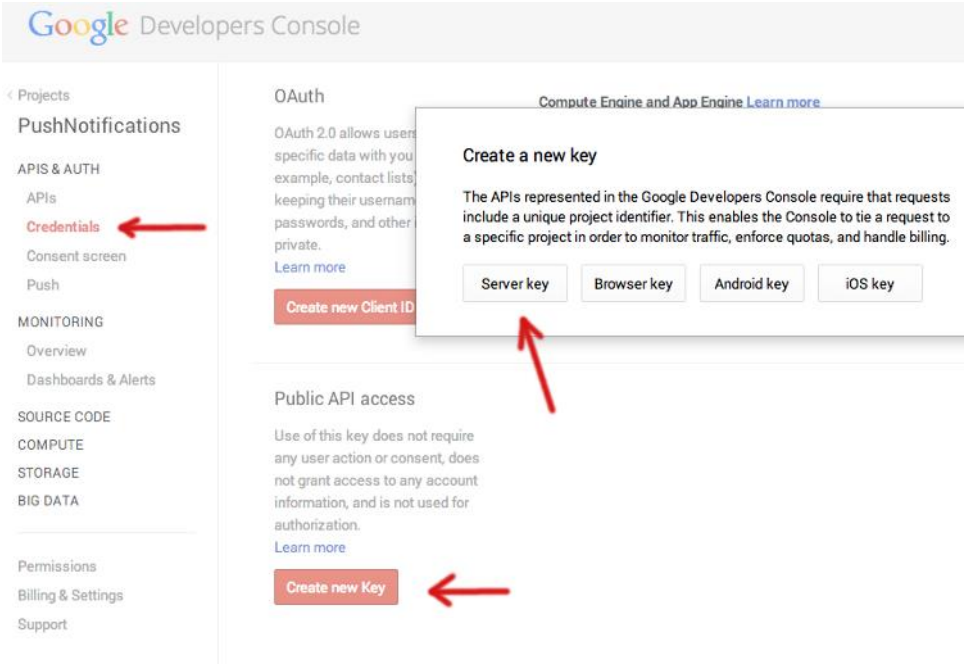

Figura 11: Crear nueva key

Y con esto obtendremos la clave de acceso a nuestra API que será utilizada por nuestro servicio.

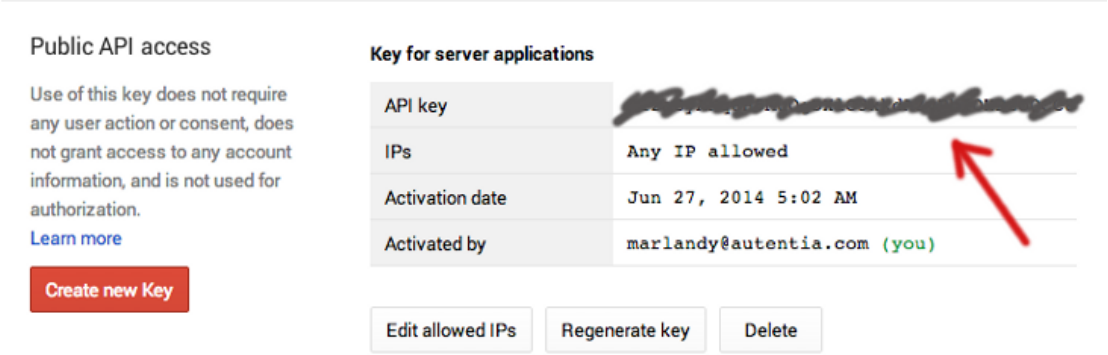

Figura 12: Obtención de la key

AIveCyANz3H5MuOgDWrC6xkdECqRbIOHBzOQDC6

Por tanto tendremos en nuestro poder el server API Key y el sender id, que utilizaremos en nuestra aplicación Nimbees para poder enviar los mensajes.

Condiciones que debe cumplir nuestra aplicación.

Para que nuestra aplicación Android pueda recibir notificaciones Push, debe cumplir con dos condiciones:

- Tener configurados los Google Play Services

- Tener habilitados una serie de permisos

Configurar Google Play Services.

Antes de habilitar Google Cloud Messaging debemos asegurarnos de que instalados los Google Play Services en nuestra SDK. Para ello, en nuestro Android SDK Manager, en la sección "Extras" podemos instalarlos en el caso de que lo hubiéramos hecho ya.

Habilitar los permisos necesarios.

Además de los servicios de Google Play, nuestra aplicación necesitará una serie de permisos para poder enviar y recibir notificaciones. Tales permisos se configuran en el fichero AndroidManifest.xml y son los siguientes:

- com.google.android.c2dm.permission.RECEIVE: permiso para que la aplicación se pueda registrar en GCM y recibir mensajes.

- android.permission.INTERNET: aunque no es obligatorio si que es muy recomendable. Gracias a este permiso podremos enviar el identificador de registro que hemos obtenido de GCM a nuestro servidor.

- android.permission.GET_ACCOUNTS: este permiso es necesario para versiones de Android inferiores a 4.0.4. Es necesario ya que GCM requiere una cuenta de Google.

- android.permission.WAKE_LOCK: permiso para mantener la pantalla apagada cuando llega una notificación.

- <paquete de la aplicación>.permission.C2D_MESSAGE: permiso para que únicamente esta aplicación pueda registrarse en GCM y recibir mensajes.

Además de habilitar los permisos citados anteriormente, necesitaremos:

- Un receptor (receiver) con el siguiente permiso habilitado com.google.android.c2dm.permission.SEND de forma que GCM pueda enviarle mensajes y con el que podremos tratarlos.

- Un servicio: al que el receptor le envía el mensaje que recibió mientras se ocupa de que el dispositivo no se quede dormido en el proceso.

Plataforma NIMBEES

NIMBEES es la primera plataforma de marketing móvil inteligente que aprovecha la información disponible en los dispositivos móviles acerca de nosotros (actividad, ubicación, preferencias…) para modelar un reflejo digital de quiénes somos.

NIMBEES utiliza esto para, mediante notificaciones push e iBeacons, permitir a las marcas realizar campañas de marketing más adecuadas y personalizadas a cada consumidor, mejorando la segmentación y las métricas para tomar estrategias adecuadas. A diferencia de otras plataformas, NIMBEES sólo almacena la información personal de cada usuario en su móvil.

NIMBEES está formada por dos partes, una web, desde la que podemos crear las aplicaciones que funcionarán con su otra parte, su API.

La web está compuesta por un menú en el que podemos ver el Dashboard, que es el panel que muestra toda la información de nuestra cuenta de usuario. En él podemos ver el número de aplicaciones registradas, el número de usuarios registrados, el número de notificaciones push enviadas y el número de notificaciones push aceptadas. Además aparece una gráfica que indica el número de notificaciones enviadas por día, junto con los últimos mensajes enviados e información de la última actividad.

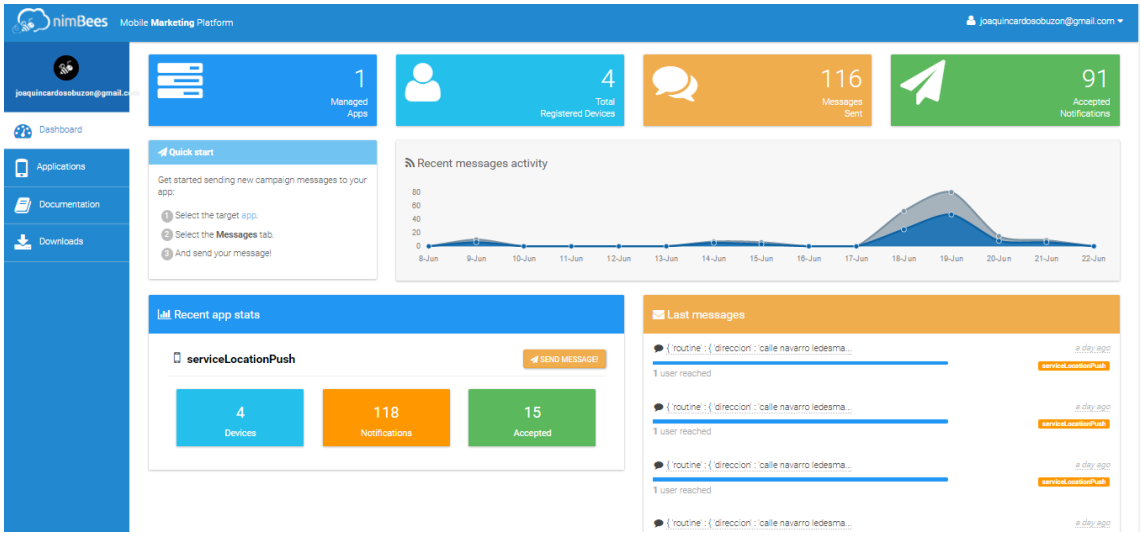

Figura 13: Dashboard Plataforma Nimbees

En la parte de Aplicationes podemos ver las aplicaciones creadas y la opción de crear una nueva aplicación.

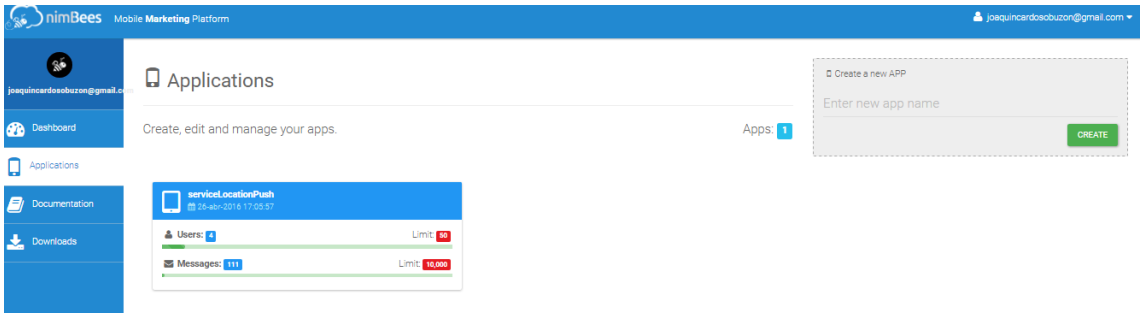

Figura 14: Aplications Plataforma Nimbees

Dentro de nuestra aplicación podemos ver información y gráficas sobre el uso de la aplicación.

En la parte de Documentación, se muestra el uso de la API e información sobre el uso de sus librerías en Android e iOS.

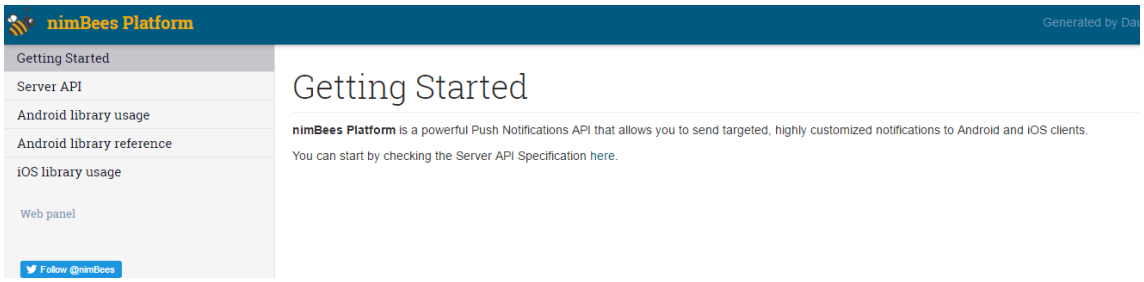

Figura 15: Documentación Plataforma Nimbees

En la parte de Downloads, se presentan las descargas de sus librerías para Android e iOS.

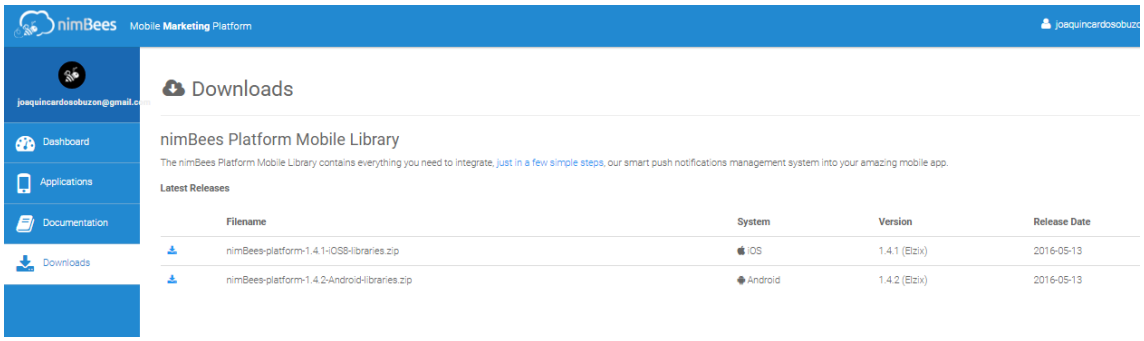

Figura 16: Librerías Plataforma Nimbees

Creación del proyecto

Dentro del menú, en Aplicaciones introducimos el nombre de nuestra app y pulsamos sobre create.

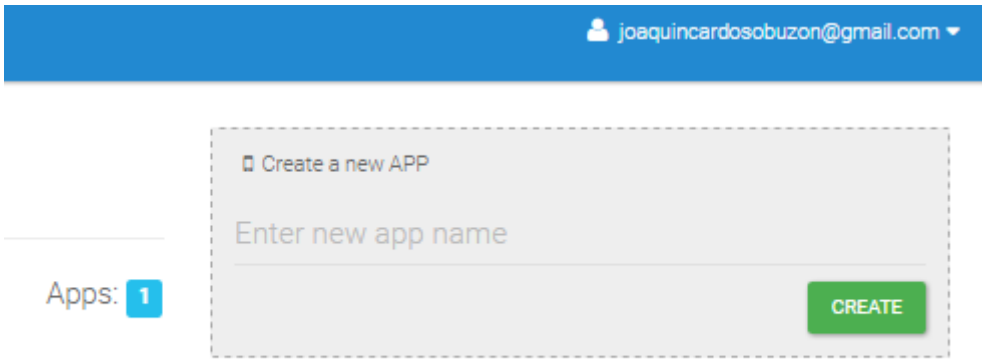

Figura 17: Nuevo proyecto

Ya tenemos nuestra app creada. Ahora pinchamos en nuestra app y aparecerá su menú de Aplications. Aquí seleccionamos App Configuration.

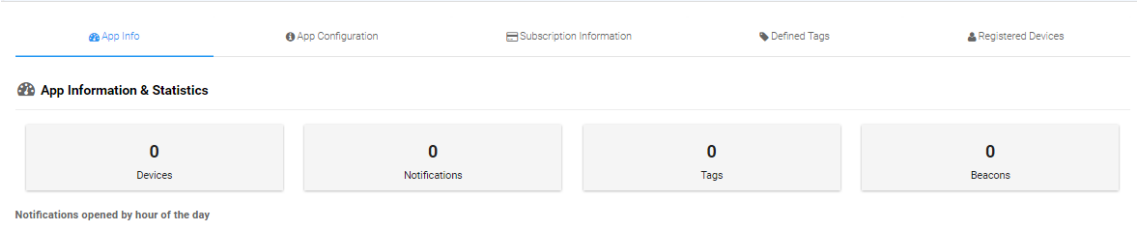

Figura 18: Menú Aplications

Aquí podremos ver información sobre nuestra app como el id de la aplicación, la fecha de creación, su nombre, el mobile key (dos claves) y las opciones de habilitar la posibilidad de enviar desde nuestra aplicación notificaciones a dispositivos Android o iOS, tan solo tendremos que seleccionar la opción deseada o ambas. Una vez hecho esto se mostrará dos entradas de texto para introducir las credenciales de Google Cloud Messages. Estas credenciales son las obtenidas al realizar el registro de nuestra aplicación en el servicio de Google, descrito en este documento.

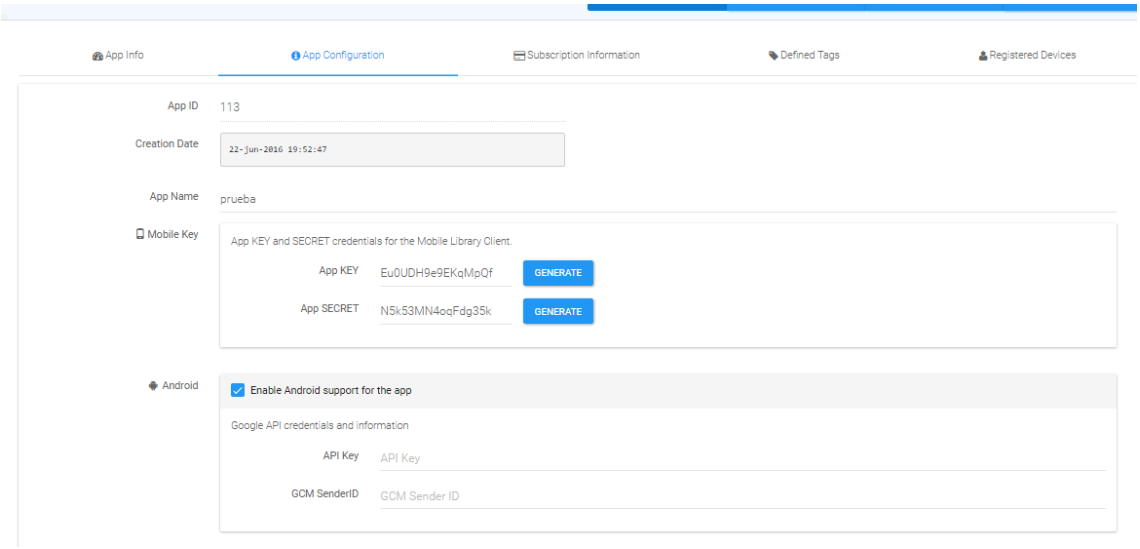

Figura 19: App Configuration

Rellenamos las credenciales obtenidas al realizar los pasos en GCM y ya tendremos nuestra aplicación lista para enviar mensajes a través de la plataforma Nimbees y GCM a nuestra aplicación móvil.

Configuración de la aplicación móvil y uso de la API de Nimbees

Para configurar nuestra app Android necesitaremos en primer lugar descargarnos las librerías descritas en la web de Nimbees anteriormente.

Importaremos el archivo *nimbees_platform_codename-x.x.x.jar* a nuestro proyecto y declararemos en el archivo build.gradle lo siguiente:

Librerias:

```
compile files('libs/nimbees-platform-deckardcain-1.3.3.jar')
compile 'com.android.support:appcompat-v7:22.2.1'
compile 'com.android.support:design:22.2.1'
compile 'com.android.support:recyclerview-v7:22.2.1'
compile 'com.android.support:support-v4:22.2.0'
compile 'com.android.support:appcompat-v7:22.2.0'
compile 'com.google.android.gms:play-services-gcm:7.0.0'
compile 'com.google.android.gms:play-services-location:7.0.0'
compile 'com.google.android.gms:play-services:7.0.0'
compile 'org.altbeacon:android-beacon-library:2.3'
compile 'com.google.code.gson:gson:2.3.1'
compile 'org.apache.commons:commons-lang3:3.4'
compile 'de.greenrobot:eventbus:2.4.0'
compile 'de.greenrobot:greendao:1.3.7'
compile 'com.squareup.okhttp:okhttp:2.4.0'
compile 'com.squareup.okio:okio:1.5.0'
compile 'com.squareup.retrofit:retrofit:1.9.0'
compile 'io.reactivex:rxjava:1.0.12'
compile 'com.squareup.picasso:picasso:2.5.2'
```

Figura 20: Librerías Nimbees

Añadimos los permisos a nuestro archivo AndroidManifest.xml

Permisos:

```
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="com.example.joaquin.servicenotificationpush.permission.C2D_MESSAGE" />
<uses-permission android:name="com.google.android.c2dm.permission.RECEIVE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
```

Figura 21: Permisos AndroidManifest.xml

Añadimos al archivo AndroidManifest.xml la versión de Google Play Services que vamos a usar:

```
<!-- Para usar servicios de Google -->
<meta-data
    android:name="com.google.android.gms.version"
    android:value="@integer/google_play_services_version"/>
```

Figura 22: Versión Google Play

Añadimos al archivo AndroidManifest.xml la actividad que será la encargada de reproducir las notificaciones recibidas, junto con los servicios.

```
<!-- Se utiliza para mostrar el contenido de las comunicaciones -->
<activity
    android:name="com.nimbees.platform.NotificationDisplayActivity"
    android:theme="@android:style/Theme.Dialog" />
```

Figura 23: NotificationDisplayActivity

Añadimos el Broadcast Receiver con el fin de procesar los mensajes de inserción y ser capaz de realizar un seguimiento de la ubicación del usuario.

```
<!-- Para procesar los mensajes de inserción y ser capaz de realizar un seguimiento de la ubicación del usuario -->
<receiver
    android:name="com.nimbees.platform.NimbeesBroadcastReceiver"
    android:enabled="true"
    android:exported="true"
    android:permission="com.google.android.c2dm.permission.SEND" >
    <intent-filter>
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />

        <category android:name="com.example.joaquin.servicecareme" />
    </intent-filter>
</receiver>
```

Figura 24: NimbeesBroadcastReceiver

Creamos un archivo denominado app.properties en la carpeta assets del proyecto, introduciendo su app key and app secret, y su identificación de remitente de GCM. Opcionalmente, también se pueden añadir los parámetros de personalización de la plataforma.

En primer lugar establemos la app.key y la app.secret, que son las que obtuvimos al crear nuestra aplicación en la plataforma Nimbees. Se encuentran dentro de “App

Configuration”. Estos parámetros se utilizarán para que se comunique nuestra aplicación móvil con la aplicación Nimbees. La primera se utiliza para autenticar y sincronizar la app móvil con la plataforma, y es una clave para uso exclusivo desde el SDK móvil. En el panel pone "Mobile Key" a la izquierda, haciendo referencia solamente a la autenticación para la librería móvil:

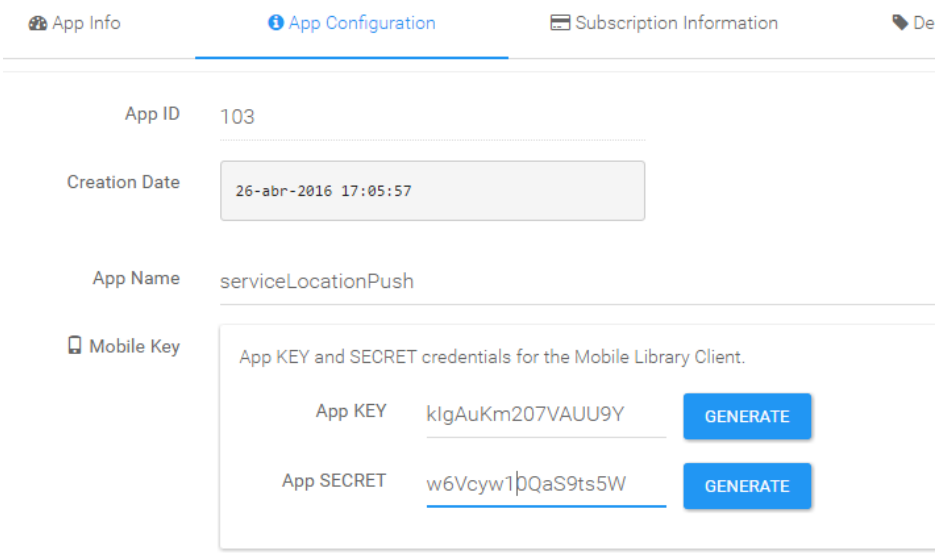

Figura 25:Mobile key

En segundo lugar establecemos el gc.sender, que es el número que obtenemos al registrarnos en google cloud messages. Este parámetro se utiliza para cuando nuestra aplicación móvil quiera enviar información a la API de Nimbees pueda enviar los mensajes a través de google cloud messages.

En tercer lugar encontramos parámetros de configuración extra como pueden ser título para la notificación, título para el diálogo, vibración, led color y parámetros de location tracking.

Por último encontramos la opción de configurar la clase CustomNotificationManager, que utilizaremos para manejar las notificaciones que recibe nuestro teléfono.
Archivo App.properties:

```
# Application key and secret
app.key=kIgAuKm207VAUU8Y
app.secret=w6VcywOOQaS9ts5W

# GCM sender id (project number)
gcm.sender=273066137809


# Mostrar más mensajes de depuración
app.nimbees.debug_mode=true

# Title of the notification [OPTIONAL]
notification.title=Notificacion nueva

# Title for the dialog [OPTIONAL]
dialog.title=Service Location Push

# Whether the notification should vibrate or not [OPTIONAL]
notification.vibration=true

# Color for the LED of the phone [OPTIONAL]
notification.led.color=#33b5e5

# Location tracking intervals [OPTIONAL]
location.tracking.interval=2700
location.tracking.fastest_interval=900

# Custom class to be used as Notification Manager [OPTIONAL - see step 9] #servicenotificationpushcationpush.CustomNotificationManager
app.notificationmanager.class=com.example.joaquin.servicenotificationpush.CustomNotificationManager
```

Figura 26: app.properties

Para gestionar las notificaciones personalizadas y mensajes de transición de pantalla, crearemos la clase CustomNotificationManager que se extiende NimbeesNotificationManager desarrollando la funcionalidad que se desee para tratar las notificaciones recibidas por Nimbees.

```
public class CustomNotificationManager extends NimbeesNotificationManager {

    public TestNotificationManager(Context context) {
        super(context);
    }

    @Override
    public void handleSimpleMessage(long idNotification, String message,  Map<String, String> additionalContent) {
        // Custom code, or just call the default implementation
        super.handleSimpleMessage(idNotification, message, additionalContent);
    }

    @Override
    public void handleCustomMessage(long idNotification, String content, Map<String, String> additionalContent) {
        // Insert your code here
    }

    @Override
    public void handleScreenTransitionMessage(long idNotification, String content, String screen, Map<String, String> additionalContent) {
        if (screen.equals(YOUR_SCREEN_NAME)) {
            showNotification(idNotification, content, additionalContent, YourActivity.class, null);
        } else {
            showNotification(idNotification, content, additionalContent);
        }
    }

    @Override
        public void handleBeaconMessage(long idNotification, String content, NimbeesBeacon beacon, String action) {
        // Insert your code here
    }
}
```

Figura 27: CustomNotificationManager

Los métodos que usaremos en nuestra aplicación son:

- handleCustomMessage: Se utiliza para tratar las notificaciones tipo “custon”, es decir, las notificaciones que utilizaremos para recibir la información adicional para tratar los filtros dentro de nuestra aplicación aplicando de este modo el modelo PeaaS.
- handleScreenTransitionMessage: Se utiliza para tratar las notificaciones de tipo “message”, que es el otro tipo de notificaciones que podemos recibir. En este tipo de notificaciones sólo se puede añadir el mensaje y una imagen. Aunque estos tipos de mensajes sólo serán procesador por nuestra aplicación móvil en caso de que la notificación push sea enviada a través de la web de Nimbees, ya que no se ha implementado esta opción en la app de envío.

Finalmente en la actividad principal de la aplicación (o la clase de aplicación personalizada), debemos inicializar la biblioteca mediante una llamada al método init del cliente nimBees:

```
NimbeesClient.init(this);
```

Después de que los usuarios se registren en la aplicación, llamamos al método RegisterDevice de NimbeesClient con el fin de registrar el usuario en la plataforma nimBees.

```
NimbeesClient.getUserManager().register("username", new NimbeesRegistrationCallback() {

    @Override
    public void onSuccess() {
        // Registration was successful! :)
    }

    @Override
    public void onFailure(NimbeesException e) {
        // Registration failed :(
    }
});
```

¡Error! No hay texto con el estilo especificado en el documento. *Figura 28: Registro Nimbees*

Y ya tenemos nuestra aplicación preparada para aceptar las notificaciones push.

Envio de notificaciones

Para el envío de notificaciones se ha implementado una función que realiza una petición post a la API de Nimbees. En la siguiente imagen podemos ver su desarrollo.

```
public void enviaMensaje2 (String[] valores) throws IOException {

    StrictMode.ThreadPolicy policy = new StrictMode.ThreadPolicy.Builder()
            .permitAll().build();
    StrictMode.setThreadPolicy(policy);
    OkHttpClient client = new OkHttpClient();

    MediaType mediaType = MediaType.parse("application/json");

    RequestBody body = RequestBody.create(mediaType,
            "{\n\"content\": {\n    \"type\": \"CUSTOM\",\n    \"message\": \"{ 'routine' : { "
                    + " 'direccion' : '"+ valores[3]+"' ,'Dia' : '"+ valores[1] +"','Mes':'"
                    + valores[2]+"', 'Radio' : '"+ valores[4] +"', 'Fecha' : '"+ valores[5]
                    +"' }, 'message' : '"+ valores[0] +"' }\"\n    }\n}");

    Request request = new Request.Builder()
            .url("https://api.nimbees.com/nimbees_platform_server_api/sendNotification/")
            .post(body)
            .addHeader("authorization", "Basic a0lnQXVLbTIwN1ZBVVU4WTpDSmdZNFk3U1puMGtkUGVE")
            .addHeader("content-type", "application/json")
            .addHeader("cache-control", "no-cache")
            .addHeader("postman-token", "a8fb2124-7ebd-e7f9-95cb-247ccc68080e")
            .build();

    Response response = client.newCall(request).execute();
    System.out.println("Response: "+response);
}
```

Figura 29: Envio de mensaje a la plataforma Nimbees

Como podemos ver en primer lugar se obtiene el policy para informar de las violaciones de políticas relacionadas con los hilos y la máquina virtual.

En segundo lugar se inicia la clase OkHttpClient, que que es una librería que permite realizar peticiones HTTP en java. Creamos el MediaType para parsear nuestro body a un objeto JSON.

En tercer lugar creamos el body de nuestro request. En él introduciremos el tipo de mensaje, que en este caso es “CUSTOM”, y la rutina con la información de los filtros que se van a usar en la aplicación que recibe las notificaciones.

En cuarto lugar creamos el request en el que debemos introducir la url de la api de Nimbees, el body, con un método post y el header, que estará compuesto por la autorización que es una clave que ahora explicaremos cómo se obtiene, el content-type, que es JSON y otros parámetros obtenido al generar el request con la aplicación “Postman”[6].

Por último realizlamos la petición, con el método build() y obtenemos la respuesta.

Obtención de la Authorization

Para la obtención de la authorization en la api de Nimbees debemos obtener dos claves en la web de Nimbees, la primera se ha explicado anteriormente cómo obtenerla, es la App KEY que se encuentra en el “app configuration” de nuestra aplicación en Nimbees.

La segunda, la MASTER SECRET, es la clave a utilizar para la API REST, tal y como se indica en la página de documentación de Nimbees.

Para obtener el valor de la MASTER SECRET en el menú "Account" que hay al desplegar tu usuario en la esquina superior derecha.

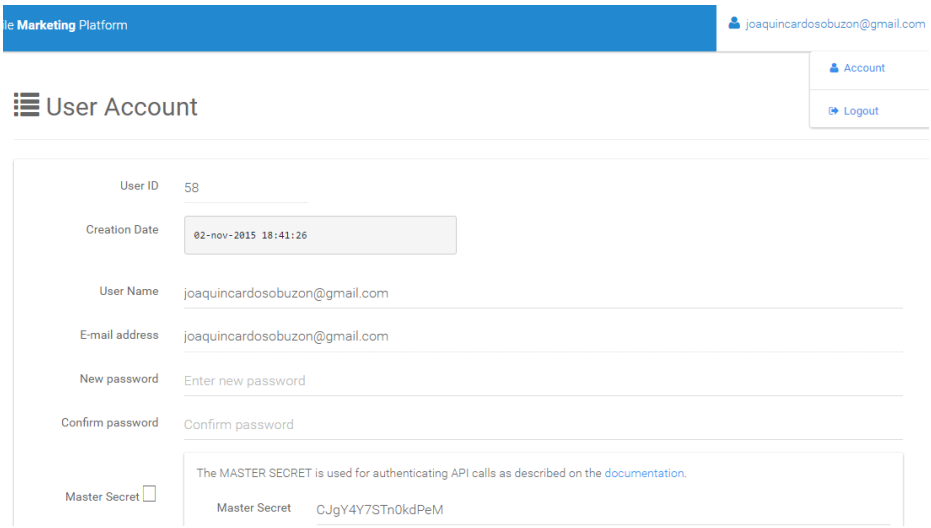

Figura 30: Master Secret

[6] Postman: Buil, test and document APIs faster. http://www.getpostman.com/

Por tanto, para la autenticación con la API hay que utilizar como nombre de usuario el App KEY, para identificar a qué aplicación de tu cuenta Nimbees y como contraseña se debe usar MASTER SECRET (en el menú Account), que es la que referencia al usuario.

Con esto ya tenemos la authorization para poder enviar mensajes a los usuarios registrados en nuestra aplicación móvil.

Filtros

Los filtros son una de las partes más importantes de nuestra aplicación, porque mediante su aplicación podremos obtener cierta aproximación de la satisfacción de condiciones obtenidas a la hora de realizar el perfil sociológico del usuario. Permitirán estudiar, si un usuario ha pasado o ha visitado una dirección, en qué día y mes fue, qué día de la semana se realizó, con qué frecuencia, etc. y mediante este estudio decidir si mostrar al usuario la notificación por presunto interés. Además estos filtros posibilitan saber si un usuario podría tener la intención de ir en un futuro a ese lugar.

Todo el proceso se realiza en la clase CustomNotificationManager que creamos para procesar las notificaciones recibidas desde la API de Nimbees. Nos centramos en el método handleCustomMessage

```
public void handleCustomMessage(long idNotification, String content, Map<String, String> additionalContent) {

    System.out.println("Llega mensaje custom"+content.toString()+ " aditional: "+additionalContent.toString());
    try {
        JSONObject jsonObj = new JSONObject(content.toString());
        Map<String,String> out = new HashMap<~>();

        Metodos.parse(jsonObj,out);

        String direccion = out.get("direccion");
        String fecha = out.get("fecha");
        String d = out.get("Dia");
        String m = out.get("Mes");
        String[] dia;
        String[] mes;
        if(d.compareTo("")!=0 && d.compareTo("[]")!=0){
            d=d.substring(1,d.length()-2);
            dia = d.split(",");
        }else {
            dia = null;
        }
        if(m.compareTo("")!=0 && m.compareTo("[]")!=0){
            m=m.substring(1,m.length()-2);
            mes = m.split(",");
        }else{
            mes = null;
        }

        String radio = out.get("Radio");
        if(radio.compareTo("")==0){
            radio = "0";
        }
        String message = out.get("message");

        FiltroNimbees f = new FiltroNimbees(context);
        if(f.buscaDireccionEnBD(direccion,fecha,dia,mes,radio)) {
            showNotification(idNotification, message, additionalContent);
        }
    } catch (JSONException e) {
        e.printStackTrace();
    }

}
```

Figura 31: Obtención del mensaje

En este método procesamos el content de la notificación que recibimos a través de la API de Nimbees. En este content, viene en forma de String, toda la información de nuestro mensaje que enviamos en formato JSON. Para poder obtener los datos, creamos el objeto JSON con el string y he creado una función que obtiene un Map<String,String> del objeto y de este modo recogemos los valores que son enviados. Hacemos las comprobaciones necesarias, por si ocurriera el caso de que no se enviara información en alguno de los campos y aplicamos el filtro. En caso de que el filtro sea aceptado se muestra la notificación, en caso contrario la rechaza.

La clase Filtro se describe a continuación. Mediante el método buscaDireccionEnBD al que se le pasan los parámetros dirección, fecha, dia, mes y radio, se va aplicando de menor a mayor exhaustividad para obtener el filtrado más preciso posible.

```
public boolean buscaDireccionEnBD(String direccion,String date,String[] dia,String[] mes,String radio){
    //Obtenemos las coordenadas de la direccion
    Double[] latLot = new Double[2];
    while(latLot[0] == null){
        latLot = getLocationFromAddress(direccion);
    }

    //Realizamos busquedas en las tablas principales
    //Si no hay fecha buscamos en la tabla Place
    List<Place> listaPlaces = db.getPlaces();
    for (Place lugar : listaPlaces) {
        if (lugar.getDescription().compareTo(direccion) == 0) {
            return true;
        }
    }

    List<RoutinePlace> listaRouPla = db.getAllRoutinePlaces();
    for(RoutinePlace lugar: listaRouPla){
        if(lugar.getPlace().getDescription().compareTo(direccion)==0){
            if(date!=null) {
                //Si existe el parametro fecha filtramos por el
                if (compruebaFecha(lugar.getEnd(),date,dia,mes)) {
                    System.out.println("Fecha encontrada");
                    return true;
                }
            }else {
                return true;
            }
    }else{
        if(radio!=null) {
            if (distFrom(lugar.getPlace().getLatitude(), lugar.getPlace().getLongitude(), latLot[0],
                    latLot[1]) <= Double.parseDouble(radio)) {
                return true;
            }
        }
    }
}


List<PlaceInstance> listaPlaceIns= db.getPlaceInstances();
for(PlaceInstance lugar: listaPlaceIns){
    if(lugar.getLatitude()==latLot[0] && lugar.getLongitude()==latLot[1]){
        if(date!=null) {
            if (compruebaFecha(lugar.getStart(),date,dia,mes)) {
                System.out.println("Fecha encontrada");
                return true;
            }
        }else {
            return true;
        }
    }else{
        if(radio!=null) {
            if (distFrom(lugar.getLatitude(), lugar.getLongitude(), latLot[0],
                    latLot[1]) <= Double.parseDouble(radio)) {
                return true;
            }
        }
    }
```

Figura 32: Filtro 1

En primer lugar recorro las tablas principales de la base de datos, que son las que deben de darme la información más reciente del usuario, estas tablas son Place, RoutinePlace y PlaceInstance. En primer lugar en cada búsqueda dentro de las tablas compruebo si la latitud longitud de la dirección recibida coincide con alguno de los puntos de estas tablas. En caso afirmativo, compruebo que se haya pasado algún parámetro de fecha, si esto es así, hago uso de un método (se explica más adelante)

que he creado para que filtre si esos lugares han sido visitados con relación a los campos pasados, que serán la fecha, los días y los meses de interés. En caso de que no coincida con ningún campo de la tabla, continúo con la siguiente buscando alguna coincidencia. Si por el contrario no se pasa fecha ni días, considero que no es necesario aplicar el filtro de la fecha y se muestra la notificación.

Si no ha habido ninguna coincidencia en ninguna de estas tres tablas, realizo la búsqueda en la tabla RoutineInstance, en este caso además de aplicar los filtros anteriores, aplico un radio si este es pasado como parámetro. Para poder aplicar el radio he definido una función, que mediante unas operaciones matemáticas obtiene la distancia de los puntos de latitud y longitud de las direcciones que estamos comparando y si la distancia es menor que el radio dado se considera que se ha tenido éxito en la búsqueda.

```
List<RoutineInstance> listaRoutIns= db.getRoutineInstances();
for(RoutineInstance lugar: listaRoutIns){
    //Como la tabla rutina instancias tiene dos direcciones compruebo las dos por si hay coincidencia
    if(lugar.getLatitude1()==latLot[0] && lugar.getLongitude1()==latLot[1]){
        if(date!=null) {
            if (compruebaFecha(lugar.getArrival(),date,dia,mes)) {
                System.out.println("Fecha encontrada");
                return true;
            }
        }else {
            return true;
        }
    }else if(lugar.getLatitude2()==latLot[0] && lugar.getLongitude2()==latLot[1]){
        if(date!=null) {
            if (compruebaFecha(lugar.getArrival(),date,dia,mes)) {
                System.out.println("Fecha encontrada");
                return true;
            }
        }else {
            return true;
        }
    }else{
        if(radio!=null) {
            //Compruebo el radio con la direccion de partida y de llegada
            if (distFrom(lugar.getLatitude1(), lugar.getLongitude1(), latLot[0], latLot[1]) <= Double.parseDouble(radio)) {
                return true;
            }
            if (distFrom(lugar.getLatitude2(), lugar.getLongitude2(), latLot[0], latLot[1]) <= Double.parseDouble(radio)) {
                return true;
            }
```

Figura 33: Filtro 2

Por último, se comprueba en la última tabla que puede contener información de todos los puntos visitados posibles. Esta tabla es la del Historial. En primer lugar comprueba si coinciden los puntos de latitud y longitud y en caso contrario se comprueba si están dentro de un radio permitido. Continuando con los filtros de fecha anteriormente mencionados.

```
List<History> listaHistory = db.getHistory();
for(History lugar: listaHistory){
    if(lugar.getLatitude()==latLot[0] && lugar.getLongitude()==latLot[1]){
        if(date!=null) {
            if (compruebaFecha(lugar.getDate(),date,dia,mes)) {
                System.out.println("Fecha encontrada");
                return true;
            }
        }else {
            return true;
        }
    }else{
        if(radio!=null && radio!="") {
            if (distFrom(lugar.getLatitude(), lugar.getLongitude(), latLot[0], latLot[1]) <= Double.parseDouble(radio)) {

                if(date!=null) {
                    if (compruebaFecha(lugar.getDate(),date,dia,mes)) {
                        System.out.println("Fecha encontrada");
                        return true;
                    }
                }else {
                    return true;
                }
            }
        }
    }
}
return false;
```

Figura 34: Filtro 3

El filtro aplicado a la fecha consiste en tres pasos, primero se comprueba si coinciden la fecha pasada en el mensaje con la de la rutina.

Si no se ha obtenido ninguna coincidencia, se pasa a comprobar los días pasados en el mensaje. Para esto se obtiene el mes en el que se produjo la rutina, si existe alguna coincidencia con los meses pasados en el mensaje, se devuelve true, en caso contrario se procede a realizar el mismo proceso, pero con los días, en este caso obtenemos el día de la semana en la que se produjo la rutina, si coincide con alguno de los días pasados en el mensaje se devuelve true. Este filtro es vital a la hora de determinar si en un futuro existe la posibilidad de que el usuario vuelva a ir a ese sitio en determinado ya que ha estado allí ese mismo día de la semana en el pasado.

En caso de que no exista ninguna coincidencia se devolverá false, para pasar al siguiente filtro.

```
public boolean compruebaFecha(Date lugar,String date,String[] dia,String[] mes){
    java.util.Date d =lugar;
    Date st = null;
    try {
        //Primero: comprobamos la fecha completa
        if(date!=null) {
            st = dateFormatDB.parse(d.toString());
            dateFormat.applyPattern("yyyy-MM-dd");
            String fecha = dateFormat.format(st);
            if (fecha.compareTo(date) == 0) {
                System.out.println("Fecha encontrada");
                return true;
            }
        }
        //Segundo: comprobamos el mes
        if(mes!=null) {
            String mesAno = mes(lugar);
            for(String m : mes) {
                if (m.compareTo(mesAno) == 0) {
                    System.out.println("Coincide mes");
                    return true;
                }
            }
        }
        //Tercero: comprobamos el dia de la semana
        if(dia!=null) {
            String diaSemana = diaDeLaSemana(lugar);
            for(String di : dia) {
                if (di.compareTo(diaSemana) == 0) {
                    System.out.println("Coincide dia");

                    return true;
                }
            }
        }
    } catch (ParseException e) {
        e.printStackTrace();
    }
    return false;
```

Figura 35: Comprueba Fecha

-Exportación / Importación de datos

Como funcionalidad extra, se pensó que sería interesante la posibilidad de que el usuario pudiera manipular la información que el sistema almacenaba dentro de su dispositivo. De este modo el usuario podría introducir nuevas rutinas y lugares, para de este modo recibir información referente a notificaciones que en primera instancia no deberían de ser recibidas por su teléfono, pero por algún tipo de motivo pudiera interesarle.

3.4 - Estructura de Datos

Para el desarrollo de la estructura de datos que manejará el sistema, se han seguido las pautas dictadas por el modelo PeaaS, que indican que toda la información acerca de los usuarios y su contexto deberían residir en su teléfono móvil, y no en un servidor externo, y es desde su dispositivo desde donde esta información es ofrecida al exterior.

Los datos son almacenados en una base de datos SQLite en el propio teléfono. SQLite es un motor de bases de datos relacional. En lugar de un sistema cliente-servidor, la biblioteca SQLite se enlaza con el programa pasando a ser parte integral del mismo. El programa utiliza la funcionalidad de SQLite a través de llamadas simples a subrutinas y funciones. Esto reduce la latencia en el acceso a la base de datos, debido a que las llamadas a funciones son más eficientes que la comunicación entre procesos. El conjunto de la base de datos (definiciones, tablas, índices, y los propios datos), son guardados como un sólo fichero estándar en la máquina host. Este diseño simple se logra bloqueando todo el fichero de base de datos al principio de cada transacción. Además Android ya incorpora todas las herramientas necesarias para su manejo, proporcionando una API bastante completa.

3.5 Optimización de recursos

Para el diseño de aplicaciones móviles la optimización del consumo de recursos debe ser una de las consideraciones más importantes. Los recursos deben ser optimizados de tal manera que no excedan en consumo de memoria del dispositivo, CPU, RAM y batería. Aunque hoy en día y para el futuro, los dispositivos móviles van aumentando su CPU y RAM, debemos intentar siempre que el consumo de estos componentes sea el menor posible, porque de esta manera conseguiremos reducir considerablemente el consumo del tercer elemento, la batería. Cuanto mayor esfuerzo realicen la CPU y la RAM mayor consumo habrá de batería, por lo que siempre hay que tenerlo en cuenta.

En primer lugar, el recibo de notificaciones push no tiene prácticamente consumo en el dispositivo, ya que se realiza

En segundo lugar, teniendo en cuenta el tamaño limitado de la memoria de los dispositivos, se ha optado por no guardar el historial de desplazamientos del usuario por un plazo indefinido. Como ya se ha mencionado anteriormente, una vez que los datos correspondientes al último día de actividad han sido procesados e integrados a la línea de tiempo del usuario podrían eliminarse. Otro caso es el de los datos intermedios generados en el análisis, podría ser útil mantener algunos y solo ir eliminando los más antiguos, ya que se trata de las muestras, pero ya tratadas y tendrán un menor impacto en la memoria. Gracias a esto puede implementarse una recuperación ante fallos en casos de que las rutinas se borren o estropeen para volver a generarlas.

La capacidad de procesamiento, el punto clave estará probablemente en el procesamiento de la actividad de la jornada, el reconocimiento de rutinas y su integración en la línea de tiempo, es decir, en el análisis, que es una tarea que consume muchos ciclos de reloj. Por ello esta actividad se lleva a cabo sólo en momentos en los que el Smartphone este siendo lo menos usado posible. Estos momentos son como hemos explicado, cuando se detecta que el teléfono está en carga, conectado a la red eléctrica, o en altas horas de la madrugada, en la que el usuario estará durmiendo.

Esta tarea, por extensión, también resulta crítica para el último recurso a tener en cuenta del dispositivo, la batería. Es por esto por lo que se decidió dar prioridad entre estas dos opciones a la de mientras esté cargando, ya que la ejecución de la tarea en este momento, no tendría repercusión en la duración de la batería.

Por último, todas las actividades de registro de localización física y monitorización también consumen mucha batería y capacidad de procesamiento durante el día. Sin embargo, en este caso resulta difícilmente evitable. Si se desea mucha precisión en los registros de información y en los cálculos de cercanía se requieren cadencias de registro muy elevadas.

3.6 – Proceso de desarrollo

Para poner en práctica el modelo PeaaS, se ha decidido crear una aplicación móvil desarrollada para el sistema operativo Android. Se ha decidido utilizar la plataforma Android porque es la plataforma más usada y que ofreces mayores posibilidades sin ningún coste. También se ha tenido en cuenta que el modelo está enfocado en dispositivos móviles, por lo que ya tenemos cubiertas las primeras condiciones.

En primer lugar obtuve el proyecto CareMee, del que se ha reutilizado la funcionalidad de monitorización del usuario y empecé a estudiar cómo implementar la funcionalidad pactada con mi tutor. Para ello busqué información para comprender mejor el modelo PeaaS y estudié el código para la implementación de la monitorización. Una vez lo tuve todo claro, establecí los requisitos funcionales y no funcionales mínimos que debía cumplir la aplicación según lo pactado. Para ello tuve en cuenta lo que se buscaba que ofreciera esta aplicación. La idea principal del proyecto, era la implementación de una aplicación móvil que fuera capaz de realizar un perfil sociológico del usuario. Para ello la monitorización implementada debía ser capaz de almacenar información de los lugares visitados por el usuario a lo largo de sus salidas, almacenar sus puntos de partida y llegada, reconocer recorridos, reconocer lugares visitados y su frecuencia, etc. También la aplicación debía de poder comunicarse con la plataforma Nimbees para poder recibir notificaciones push. Además se consideró la posibilidad de poder volcar la información almacenada en la base de datos de la aplicación en un archivo Excel y txt, para que el usuario tuviera la posibilidad de ver la información obtenida y poder manipularla y volver a volcar los cambios a la base de datos. De este modo le ofrecemos la posibilidad de poder administrar su información de la manera que el

considere más interesante, ya que según esta información, se obtendrán las notificaciones push que se recibirán a través de la plataforma Nimbees. Las notificaciones se decidieron que se obtuvieran y se procesaran en el dispositivo móvil por medio de una serie de filtros que serían los encargados de decidir si se debían mostrar o no. Los filtros, debían, mediante unos parámetros recibidos en la notificación, estudiarían la base de datos y decidirían si la notificación sería mostrada al usuario o no. Estos filtros también deberían permitir intentar adivinar si según los parámetros pasados y los que se encuentran en la base de datos de fecha y posicionamiento por ejemplo, si el usuario pudiese estar interesado en un futuro, para mostrar la notificación.

Después de la obtención de esa funcionalidad, se procedió a la creación del proyecto e implementar la monitorización obtenida según lo establecido en los requisitos. Para ello se hizo uso de la funcionalidad de CareMee y se adaptó ya que el funcionamiento es distinto, la aplicación CareMee, está orientada a pacientes con Alzheimer. Esta app se encarga de reconocer recorridos habituales de los pacientes, avisando a los cuidadores en caso de que el paciente salga de un recorrido habitual.

En el siguiente paso, realicé el registro en la web de Nimbees y la creación de la aplicación en la web. Seguidamente realicé el proceso de registro en Google Cloud Messages y obtuve las claves para poder utilizar este servicio junto con la aplicación de Nimbees. Una vez obtuve las claves las introduje en la app de Nimbees para permitir la comunicación.

Cuando ya tenía todo funcionando, procedí a comenzar el desarrollo de la funcionalidad necesaria para permitir que la app permitiera registrarse y recibir mensajes desde la api de Nimbees.

Para el envío de mensajes, pensamos en realizar una app nueva donde implementar esta funcionalidad, ya que ésta no es considerada para la principal, porque no se considera necesaria. Para el desarrollo se realizó una aplicación simple, que recogiera una serie de datos, que son la información que se usarán como filtro en la aplicación principal, y se enviarán a la api de Nimbees, que reenviará el mensaje a la aplicación principal. Para generar la petición post que enviaremos a Nimbees, utilicé una aplicación llamada Postman, que es una aplicación que en nuestro caso corre como una app de google Chrome, y se utiliza para generar peticiones post. En primer lugar se genera un archivo con formato JSON, donde se pasan todos los parámetros necesarios para realizar la petición, se muestra en el siguiente un ejemplo de dicho archivo:


```
{
    "id": "1dbb8d07-ad05-4d86-3e08-4c4019b771fc",
    "name": "nimBees Server API",
    "description": "",
```

```
        "order": [
                "f4d2fb88-3304-5574-add6-e2612546cd64",
                "c6ee8f43-5668-68ed-5053-8798972405b3"
        ],
        "folders": [],
        "timestamp": 1461830224735,
        "owner": "146062",
        "remoteLink": "",
        "public": false,
        "requests": [
                {
                        "id": "c6ee8f43-5668-68ed-5053-8798972405b3",
                        "headers": "Authorization: Basic
Sks5STZlR1NvM1JrTDM3cTpPcUw0c3ZsbTNIWlBNZTWw\nContent-Type:
application/json\n",
                        "url":
"https://api.nimbees.com/nimbees_platform_server_api/getNotificationStatus/33720
",
                        "preRequestScript": null,
                        "pathVariables": {},
                        "method": "GET",
                        "data": [],
                        "dataMode": "raw",
                        "version": 2,
                        "tests": null,
                        "currentHelper": "basicAuth",
                        "helperAttributes": {
                                "id": "basic",
                                "username": "kIgAuKm207VAUV8Y",
                                "password": "CJgY4Y7SZN0kdPeD",
                                "saveToRequest": true
```

```
        },
        "time": 1464198034694,
        "name": "getNotificationStatus/<id>",
        "description": "",
        "collectionId": "1dbb8d07-ad05-4d86-3e08-4c4019b771fc",
        "responses": [],
        "rawModeData": "{\n\"content\": {\n    \"type\":
\"MESSAGE\",\n    \"message\": \"Hola!\"\n    }\n}"
    },
    {
        "id": "f4d2fb88-3304-5574-add6-e2612546cd64",
        "headers": "Authorization: Basic
Sks5STZlR1NvM1JrTDM3cTpPcUw0c3ZsbTNIWlBNZTWw\nContent-Type:
application/json\n",
        "url":
"https://api.nimbees.com/nimbees_platform_server_api/sendNotification/",
        "preRequestScript": null,
        "pathVariables": {},
        "method": "POST",
        "data": [],
        "dataMode": "raw",
        "version": 2,
        "tests": null,
        "currentHelper": "basicAuth",
        "helperAttributes": {
            "id": "basic",
            "username": "kIgAuKm207VAUU8Y",
            "password": "CJgY4Y7SZn0kdPeD",
            "saveToRequest": true
        },
        "time": 1464198040085,
```

```
                    "name": "sendNotification/",
                    "description": "",
                    "collectionId": "1dbb8d07-ad05-4d86-3e08-4c4019b771fc",
                    "responses": [],
                    "rawModeData": "{\n\"content\": {\n    \"type\":
\"MESSAGE\",\n    \"message\": \"Hola!\"\n    }\n}",
                }
        ]
}
```


Como podemos observar, se incluye el formato de una petición post y otra get, y se le pasan, entre otras cosas, la clave de autenticación que es la Master secret de google cloud messages utilizada como autenticación de acceso básica, el usuario y la password de que son la App Key y la Master Secret de la cuenta de Nimbees, la url a la que se realizará la petición y el mensaje donde se enviará toda la información para los filtros.

Importamos dicho archivo a la aplicación Postman y gracias a ella agilizamos el proceso de comunicación. Una vez conseguimos que la api nos respondiera un mensaje de “OK”, se procedió a transformar la petición para java. Para ello Postman ofrece la posibilidad de transformar el objeto JSON en una petición para el lenguaje que nosotros prefiramos, que en nuestro caso es java.

Una vez podíamos enviar y recibir las peticiones en nuestro teléfono, se procedió a implementar los filtros para decidir si la aplicación debía mostrar o rechazar el mensaje.

4- Conclusiones y trabajos futuros.

Como se describe a lo largo de este proyecto se pretende presentar una tecnología que ofrece al propio usuario ser quien maneja la información que su dispositivo almacena sobre sí mismo. Esta situación podría mejorar la información que el teléfono va a mostrarle según su perfil creado, como por ejemplo mensajes publicitarios o de interés.

El modelo PeaaS probablemente no sea de alto interés para los sistemas publicitarios ya que el interés de éstos es llegar al mayor número posible de usuarios, sin embargo, si puede llegar a ser interesante para los usuarios debido a que ellos pueden filtrar y manipular su información o perfil para recibir sólo aquello que puede ser de su interés. De este modo se evita que el usuario sea bombardeado con multitud de información que probablemente no sea de su interés.

Para futuros trabajos se podrían establecer nuevos filtros que permitan seleccionar mejor la población que puede estar interesada. También se podría mejorar el almacenamiento de los lugares que visita el usuario, por ejemplo en el actual sistema se almacena una dirección, cosa que se puede mejorar porque por ejemplo si se visita un centro comercial no se tiene en cuenta qué tiendas o tipo de tiendas son visitadas. También se podría permitir la posibilidad de que el usuario introduzca información sobre sí mismo, como pueden ser cosas de interés, gustos deportivos, aficiones, etc. De este modo si al usuario le gusta el running por ejemplo, y se va a celebrar una carrera deportiva cerca de donde vive, o en su comunidad autónoma, pueda recibir dicha información. Podría resultar interesante que se monitorizaran búsquedas y visitas de páginas que el usuario realiza desde su dispositivo, para tener un perfil aún más desarrollado.

Si el modelo PeaaS tomara fuerza y se siguiera desarrollando aplicaciones utilizando esta lógica la publicidad que reciben los usuarios podría ser más interesante, ya que ellos seleccionarán qué recibir y no recibir mucha publicidad de forma constante y desinteresada teniendo que buscar cual puede llegar a interesarle. Con este modelo se filtra la publicidad o información antes de que la reciba el usuario y no tenga que ser el usuario quien la filtre después de recibirla. Además la información del propio usuario no irá de una empresa a otra sin el consentimiento del propio usuario, cosa que hoy en día es un gran problema para los ciudadanos.

Referencias Bibliográficas:

[1] J. Guillen, J. Miranda, J. Berrocal, J. Garcia-Alonso, J.M. Murillo, C. Canal. “People as a Service: A Mobile-centric Model for Providing Collective Sociological Profiles”

[2] http://encuentracapital.es/encuentra_capital_III/nimbees/

[3] Libro Introduccion a Android www.tecnologiaUCM.es

[4] http://elbauldeandroid.blogspot.com.es/2013/12/services.html

[5] http://jexcelapi.sourceforge.net/

[6] Frances Wright. *Course of Popular Lectures*. Office of the Free Enquirer, 1829. 24

ANEXOS

5- Manual de usuario

-Manual de usuario de aplicación “Mis rutinas”.

La aplicación Mis rutinas, ofrece a sus usuarios la posibilidad de tener un perfil sociológico en su teléfono siguiendo el modelo PeaaS. A su vez permite el volcado de datos en un archivo Excel y la posibilidad de poder manipular ese Excel e importarlo de nuevo a la base de datos.

Para su uso será necesario disponer un dispositivo móvil con sistema operativo Android.

Una vez instalada la aplicación en nuestro dispositivo, no tendremos que realizar ninguna acción, tan sólo asegurarnos que el dispositivo tiene activado el gps, para que la aplicación comience su lectura de datos.

Conforme vayan pasando los días, la aplicación irá definiendo cada vez, de forma más clara el perfil del usuario. Según el usuario vaya viajando y realizando sus recorridos, la aplicación reconocerá o detectará si es un recorrido ya realizado al que habrá que aumentar su frecuencia, o si habrá que añadirlo como nuevo.

Una vez el usuario comience a recibir su primera notificación push, éstas, se irán mostrando en la actividad principal de la app durante un periodo de tiempo, de este modo nos aseguramos de que la información de las notificaciones no sean olvidadas por el usuario.

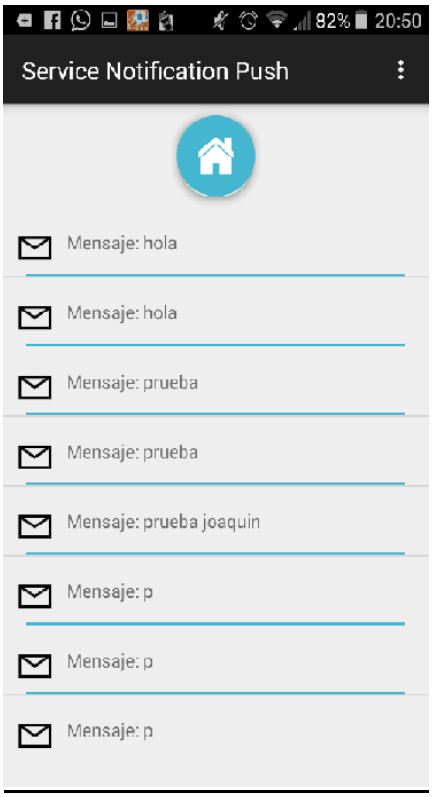

Figura 37: Vista principal aplicación

Si el usuario pulsa su dedo sobre cualquier notificación, ésta se abrirá como si fuera una alerta, mostrando su información.

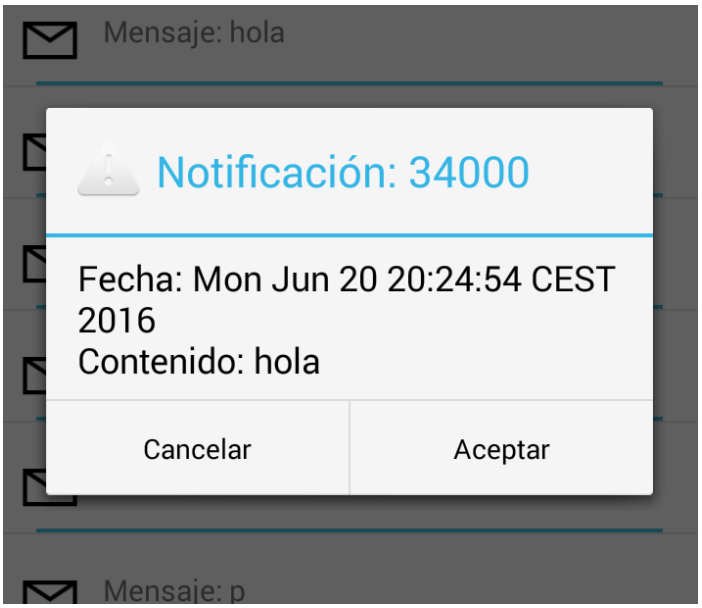

Figura 38: Vista notificación

En el menú que se encuentra en la esquina superior derecha, encontraremos opciones para exportar la base de datos a un archivo Excel, importar el archivo Excel a la base de datos, hacer un backup de la base de datos y poder recuperar de un archivo de backup la base de datos.

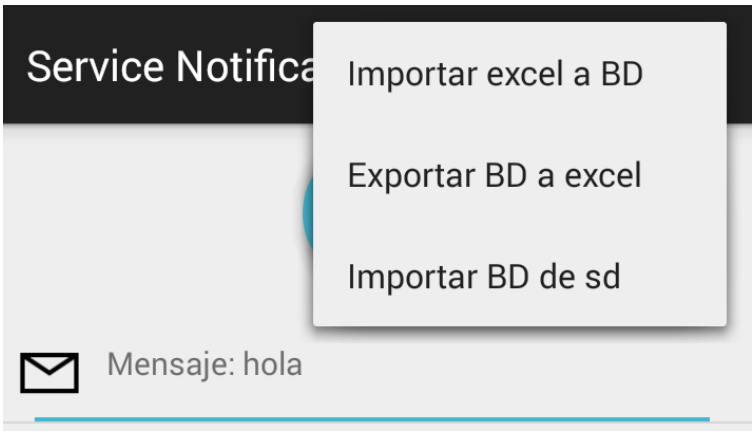

Figura 39: Menú aplicación

Las opciones de backup sobre la base de datos, están pensadas para evitar que se pierda la información si al hacer una importación del archivo Excel introdujéramos algún tipo de error, corrompiendo de este modo la base de datos.

El archivo Excel se guarda dentro de la carpeta raíz de la memoria del teléfono con nombre de serviceNotificationPush.xml y el archivo de backup serviceNotificationBackup.

El formato del archivo Excel, en primer lugar se muestra la primera fila donde podemos ver la palabra que representa al tipo de datos a mostrar, que en este caso es “Lugares:”. En la siguiente línea aparecen los nombres de las columnas que son ID, longitud, latitud, dirección y frecuencia. Para diferenciar un conjunto de datos de otro se introduce una línea en blanco. La siguiente lista de datos a mostrar es la de “Instancias:” que contiene datos de ID, latitud, longitud, dirección, localidad, fecha de inicio y fecha de fin. El último conjunto de datos es el de “Instancia-Rutina” y consta de ID, dirección de inicio, localidad de inicio, dirección de fin, localidad de fin, fecha de inicio y fecha de fin. En el caso de que no se tengan datos dentro de la aplicación sobre alguno de estos conjuntos, se mostrará una fila mostrando el mensaje de “No se encuentran Lugares”, “No se encuentran Instancias” y “No se encuentran Instancias-Rutinas” respectivamente.

A	B	C	D	E
Lugares:				
ID	Longitud	Latitude	Dirección	Frecuencia
1	-4.4792361	36.7239433	Calle Navarro Ledesma, 239, Málaga	100
7	-4.48044942	36.72390195	Av Parménides, 4, Málaga	33
8	-4.4788494	36.7230028	Calle Esquilo, 27, Málaga	21
2	-4.47871145	36.72428264	Calle Navarro Ledesma, 239, Málaga	11
3	-4.4769711	36.7211588	Av de Plutarco, 24, Málaga	6
9	-4.4799779	36.7219533	Av Parménides, 11, Málaga	5
4	-4.41373087	36.71626991	Paseo del Muelle Dos, Málaga	4
5	-4.41338759	36.71736888	Paseo del Muelle Dos, Málaga	4
6	-4.42264814	36.7173328	Alameda Principal, 19, Málaga	4

Instancias:						
ID	Longitud	Latitude	Dirección	Localidad	Inicio	Fin
155	-5.7439	36.815149	Calle Clérigos, 2	Bornos	2016-06-25 12:15:59	2016-06-25 14:59:31
154	-5.745016	36.814087	Calle Alta, 34	Bornos	2016-06-25 11:33:31	2016-06-25 12:15:59
153	-5.16525	36.746639	Calle de José María Castelló Madrid, 20	Ronda	2016-06-25 09:47:42	2016-06-25 10:06:29
152	-4.479645	36.724479	Calle Navarro Ledesma, 166	Málaga	2016-06-22 17:10:38	2016-06-22 18:36:57
151	-4.479231	36.723462	Calle Esquilo, 28	Málaga	2016-06-22 16:50:33	2016-06-22 17:10:38
150	-4.479491	36.724830	Calle Navarro Ledesma, 166	Málaga	2016-06-22 15:29:56	2016-06-22 16:50:33
149	-4.4966	36.702539	Calle la Gitanilla, 23-25	Málaga	2016-06-22 08:02:49	2016-06-22 15:07:33
148	-4.480325	36.723488	Calle Esquilo, 26	Málaga	2016-06-22 06:26:05	2016-06-22 07:48:16
147	-4.479150	36.724112	Calle Navarro Ledesma, 241	Málaga	2016-06-22 03:09:39	2016-06-22 06:26:05
146	-4.47950	36.724000	Av Parménides, 1	Málaga	2016-06-21 23:09:31	2016-06-22 03:02:00
145	-4.479463	36.724003	Calle Navarro Ledesma, 243	Málaga	2016-06-21 20:26:40	2016-06-21 22:59:29
144	-4.479491	36.724016	Calle Navarro Ledesma, 243	Málaga	2016-06-21 18:34:05	2016-06-21 20:26:18
143	-4.479472	36.723987	Av Parménides, 1	Málaga	2016-06-21 00:35:56	2016-06-21 07:48:40
142	-4.479107	36.723966	Calle Navarro Ledesma, 241	Málaga	2016-06-20 23:12:39	2016-06-20 23:51:59
141	-4.480345	36.724740	Calle Navarro Ledesma, 168	Málaga	2016-06-20 22:52:20	2016-06-20 23:12:39

Instancias-Rutinas:						
ID	Dirección Inicio	Localidad Inicio	Dirección Fin	Localidad Fin	Inicio	Fin
1	Calle Franz Kafka	Málaga	Calle Píndaro, 10	Málaga	2016-02-24 19:56:10	2016-02-24 20:21:54
2	Bulevar Louis Pas	Málaga	Av Parménides, 1	Málaga	2016-03-09 17:55:28	2016-03-09 18:11:31
3	Av Parménides, 1	Málaga	Calle Navarro Ledesma, 241	Málaga	2016-03-18 20:22:56	2016-03-18 20:47:22
4	Calle Navarro Led	Málaga	Calle Navarro Ledesma, 241	Málaga	2016-03-18 21:44:18	2016-03-18 22:19:14
5	Calle Navarro Led	Málaga	Urbanización Arcos Elviria Pueblo, 45	Marbella	2016-03-20 11:56:36	2016-03-20 12:49:19
6	Urbanización Arc	Marbella	Urbanización Arcos Elviria Pueblo, 45B	Marbella	2016-03-20 13:05:10	2016-03-20 13:21:41
7	Urbanización Arc	Marbella	Urbanizacion Elviria Sur, 30	Marbella	2016-03-20 13:38:39	2016-03-20 14:51:16
8	Urbanizacion Elvi	Marbella	Av Parménides, 11	Málaga	2016-03-20 15:51:25	2016-03-20 20:12:08
9	Calle Navarro Led	Málaga	Calle Navarro Ledesma, 241	Málaga	2016-03-20 21:54:08	2016-03-20 22:20:06
10	Calle Navarro Led	Málaga	Calle Esquilo, 27	Málaga	2016-03-28 23:37:23	2016-03-29 00:18:31
11	Calle Esquilo, 27	Málaga	Calle Nazarín, 4	Málaga	2016-03-29 08:40:59	2016-03-29 09:43:21
12	Carretera de la Az	Málaga	Calle Navarro Ledesma, 243	Málaga	2016-03-29 14:41:47	2016-03-29 15:20:33
13	Calle Esquilo, 27	Málaga	Calle Esquilo, 27	Málaga	2016-03-29 22:59:20	2016-03-29 23:18:26
14	Calle Esquilo, 27	Málaga	Calle Esquilo, 27	Málaga	2016-03-30 19:41:50	2016-03-30 20:15:28
15	Calle Esquilo, 27	Málaga	Calle Esquilo, 27	Málaga	2016-03-30 20:51:28	2016-03-30 21:50:17
16	Calle Esquilo, 27	Málaga	Calle Esquilo, 27	Málaga	2016-03-30 22:30:07	2016-03-30 22:56:05

Figura 40: Ejemplo de archivo excel

El usuario tiene la posibilidad de manipular la información almacenada en su teléfono, pero para ello se debe mantener el formato de los valores que se vuelcan de la base de datos. Para evitar que ocurra algún tipo de error, siempre es recomendable hacer un backup de la base de datos antes de realizar cualquier tipo de importación del archivo Excel a la base de datos, ya que para esto se ha implementado dicha funcionalidad. Para ello el usuario tan sólo tendrá que pulsar sobre dicha opción en el menú y la aplicación se encargará de realizar de obtener el archivo y realizar todo el proceso.

-Manual de usuario aplicación de "Envío de notificaciones".

La aplicación de envío es una aplicación muy simple que consta de una serie de entradas de texto, entre las que se incluyen el mensaje, la fecha, dirección, radio, un checkbox para los días y otro para los meses.

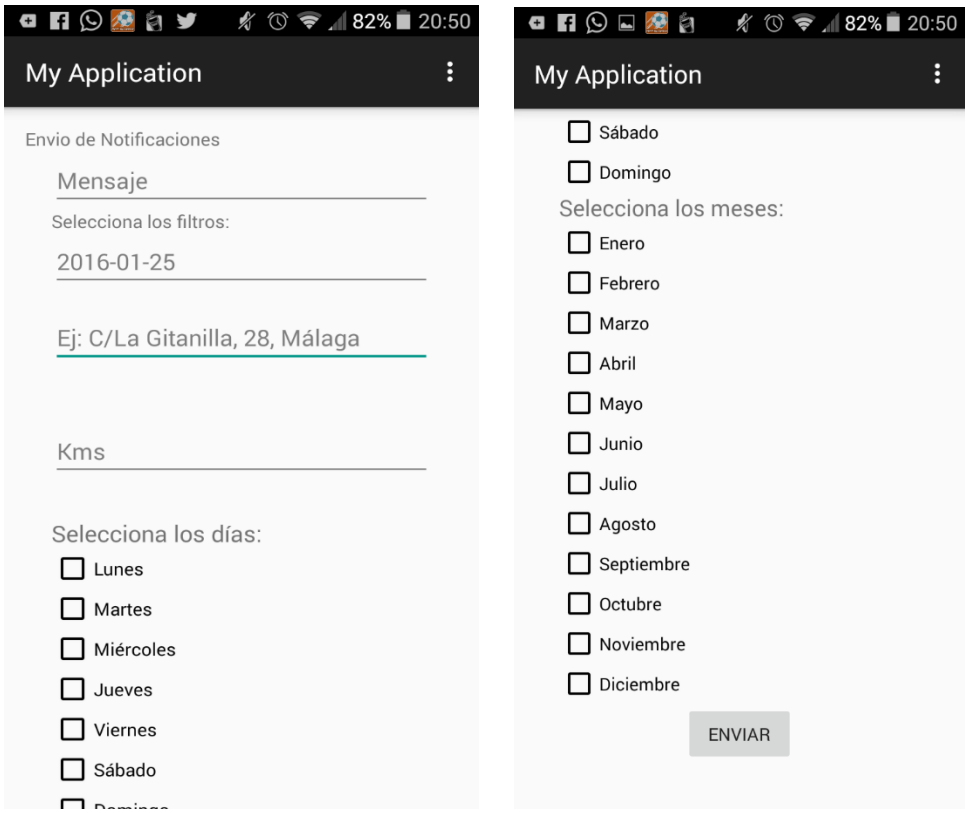

Figura 41: Vista aplicación

El usuario tiene la posibilidad de elegir que enviar, una vez elegido los datos, el usuario únicamente tiene que pulsar sobre el botón Enviar y la notificación será recibida en los teléfonos registrados dentro de la aplicación en la plataforma Nimbees.

6- Casos de uso

A continuación se presentan casos de uso que ponen en práctica el funcionamiento del sistema elaborado.

6.1- Caso de uso 1.

Para el primer caso de uso se emulará una situación en la que un usuario envía una nota publicitaria para que sea mostrada en los dispositivos que cumplan los requisitos especificados.

En primer lugar partimos de un usuario que quiere publicitar una fiesta en su local, su intención es que esta nota llegue a aquellos usuarios que frecuentan ir a su local los jueves del último mes. También está interesado en que las personas que hayan ido a locales cercanos reciban su nota publicitaria porque son posibles interesados que pasan habitualmente por su local.

El usuario hace uso de la aplicación “Envío de notificación”. En dicha aplicación se introducen los datos por los que el usuario quiere que se utilicen para filtrar, es decir, los datos que utilizara la aplicación “Mis rutinas” para filtrar y decidir si mostrar o no la notificación. Estos datos son:

-El mensaje o nota publicitaria.

-La dirección del lugar.

-Una fecha.

-Día de la semana.

-Mes del año.

-Radio de alcance cercano al lugar.

El usuario seleccionará los datos que crea convenientes para que su notificación llegue a los dispositivos que más le interese, de este modo si por ejemplo el usuario no introduce una fecha o día de la semana, estos filtros no serán aplicados en la aplicación “Mis rutinas”.

Una vez enviado el mensaje, éste llegará a la plataforma Nimbees, donde existe una aplicación que será la encargada de enviar a través de Google Cloud Messages a todos los dispositivos registrados en la aplicación Nimbees. Cuando la aplicación Nimbees reenvía el mensaje, éste es recibido en el dispositivo de cada usuario en la aplicación “Mis rutinas”. Al recibir el mensaje, la aplicación aplica los filtros necesarios para evaluar si según los datos que existen en el mensaje, el contenido es del interés de su usuario propietario o no. Para ello estos filtros harán uso de los datos que ha ido recolectando a través del tiempo de los lugares que habitualmente visita su usuario, en qué fecha lo hizo, llegando a poder predecir si en algún momento futuro pudiese volver a ir.

6.2- Caso de uso 2.

En este caso de uso se mostrará el acceso que tiene el usuario de sus datos y cómo puede llegar a manipularlos.

Para ello partimos de un dispositivo que lleva corriendo la aplicación “Mis rutinas” durante algún tiempo y tiene datos suficientes recogidos en su terminal.

En primer lugar el usuario, exportará su base de datos a un archivo Excel, haciendo uso de una opción del menú de la aplicación “Mis rutinas”. Una vez terminada la exportación de su base de datos, aparecerá un archivo en la carpeta raíz de su dispositivo. Al abrir este archivo se verá una descripción como en la siguiente imagen:

Lugares:						
ID	Longitud	Latitude	Dirección	Frecuencia		
2	-6.3717017	39.4772317	Calle San Justo, 15, Cáceres	11		
1	-6.3720217	39.4781233	Plaza 8 de Septiembre, 9, Cáceres	4		
Instancias:						
ID	Longitud	Latitude	Dirección	Localidad	Inicio	Fin
33	-3.699136	40.411398	Calle de Santa Isabel, 16	Madrid	2016-06-26 23:21:28	2016-06-26 23:36:57
32	-3.697131	40.412656	Calle San José, 8	Madrid	2016-06-26 22:54:54	2016-06-26 23:10:10
31	-3.7031	40.407516	Calle Tribulete, 21	Madrid	2016-06-26 19:54:59	2016-06-26 20:19:23
13	-6.375013	39.465621	Calle Madre Isabel Larrañaga, 1	Cáceres	2016-06-23 13:46:47	2016-06-23 16:27:44
12	-6.340568	39.479013	Av de las Letras	(Campus Universitario)	2016-06-23 11:45:20	2016-06-23 13:29:02
11	-6.3426	39.4793	Av de las Letras	(Campus Universitario)	2016-06-23 10:13:50	2016-06-23 11:18:10
10	-6.371144	39.477643	Calle San Justo, 8	Cáceres	2016-06-23 08:07:52	2016-06-23 08:35:18
9	-6.372361	39.477459	Calle General Margallo, 44	Cáceres	2016-06-23 04:23:53	2016-06-23 07:55:34
8	-6.372001	39.47726	Calle General Margallo, 26	Cáceres	2016-06-23 00:52:36	2016-06-23 04:23:49
7	-6.371906	39.4771	Calle General Margallo, 22	Cáceres	2016-06-22 22:54:26	2016-06-23 00:51:30
6	-6.277556	39.448099	Vial 2, 2-6	Sierra de Fuentes	2016-06-22 21:23:33	2016-06-22 21:40:50
5	-6.278653	39.448778	Vial 3, 5D	Sierra de Fuentes	2016-06-22 20:06:14	2016-06-22 20:42:16
4	-6.278414	39.4485	Vial 4, 17A		2016-06-22 18:09:48	2016-06-22 18:27:51
3	-6.370596	39.474244	Calle Manga, 2	Cáceres	2016-06-22 16:00:44	2016-06-22 18:00:31
2	-6.373418	39.473703	Plaza de San Juan, 18	Cáceres	2016-06-22 14:40:40	2016-06-22 14:55:43
1	-6.341458	39.47863	Calle Trujillo, 13	(Campus Universitario)	2016-06-22 11:48:48	2016-06-22 13:38:49
Instancias-Rutinas:						
ID	Dirección Inicio	Localidad Inicio	Dirección Fin	Localidad Fin	Inicio	Fin
1	Calle Trujillo, 13	(Campus Universitario)	Plaza de San Juan, 18	Cáceres	2016-06-22 13:38:49	2016-06-22 14:40:40
2	Plaza de San Juan, 18	Cáceres	Calle Manga, 2	Cáceres	2016-06-22 14:55:43	2016-06-22 16:00:44
3	Vial 4, 17A		Vial 3, 5D	Sierra de Fuentes	2016-06-22 18:27:51	2016-06-22 20:06:14
4	Vial 3, 5D	Sierra de Fuentes	Vial 2, 2-6	Sierra de Fuentes	2016-06-22 20:42:16	2016-06-22 21:23:33
5	Calle San Justo, 8	Cáceres	Av de las Letras	(Campus Universitario)	2016-06-23 08:35:18	2016-06-23 10:13:50

Figura 42: Vista documento Excel.

Como podemos observar aparecen los datos almacenados sobre: Lugares, Instancias y Rutinas. Estos son los datos que la aplicación “Mis rutinas” ha almacenado sobre él. En caso de que el usuario quiera realizar una prueba, o manipular algún dato que considere oportuno puede realizar los cambios correspondientes en el archivo Excel.

Una vez hecho los cambios, tan sólo guarda el documento y elige la opción importar archivo Excel a BD dentro del menú de la aplicación “Mis rutinas”. Una vez terminado el proceso se mostrará un mensaje al usuario confirmando la finalización de la importación.

Concretando un poco más. Imaginémonos que a un usuario que está interesado recibir posibles notificaciones que puedan ser enviadas a usuarios residentes en la ciudad de Málaga ya que la próxima semana va a pasarla allí. Normalmente no recibiría ninguna notificación en cuyos filtros aparezca la ciudad de Málaga, debido a que esta ciudad no aparecería dentro de los datos que la aplicación “Mis rutinas” ha almacenado del usuario. Entonces el usuario tan sólo tendría que añadir dentro de alguno de los apartados descritos anteriormente en el documento (Lugares, Instancias y Rutinas). Tan sólo sería necesario introducir la dirección y la fecha, la latitud y longitud se generarían automáticamente. Una vez realizada importación del archivo Excel

manipulado, ya podremos recibir las notificaciones que incluyan el lugar en el que está interesado en sus filtros.

7- Herencia

7.1- Funcionalidad heredada

En este apartado presentamos parte de la funcionalidad que se ha heredado de un antiguo proyecto de fin de grado llamado Caremee descrito anteriormente.

Acumulación / Monitorización

Este es el componente consta de dos tareas que se van realizando de forma simultánea. Estas tareas son la de acumulación de histórico y la de monitorización de los desplazamientos del usuario. Dicha funcionalidad se realiza a través de un servicio que se ejecuta en segundo plano mientras el dispositivo se encuentra encendido junto con su conexión a internet y gps activados.

La acumulación funciona con un Listener que detecta cuando un usuario recorre una distancia x, en este momento se recogen los datos del posicionamiento del dispositivo (latitud y longitud) y se almacenan en la base de datos con la fecha y la hora de la recogida. Se realiza una recogida cada cierta distancia para ahorrar en el uso de batería y memoria. De este modo, si un usuario permanece durante un largo periodo de tiempo en un determinado lugar, el Listener quedará en reposo.

Después de realizar la llamada al Listener, se lleva a cabo el proceso de monitorización, que actúa siguiendo el siguiente procedimiento: utilizamos un contador que se incrementa cada vez que se hace una llamada al Listener, en caso de que en una llamada se haga después de más de 5 minutos desde la última monitorización, se comprueba que se hayan realizado al menos un determinado número de incrementos del contador, consiguiendo el mínimo consumo de batería con un rango de detección de desvíos bastante aceptable.

Explicando un poco mejor el procedimiento, en cuanto recogemos la primera muestra del histórico (recogida de datos del gps), actúa el monitor rellenando una lista con todas las rutinas que parten del lugar de la muestra recogida e incrementando el contador. Seguidamente, comprobamos si han pasado 5 minutos, si no han pasado, se incrementa el contador, en caso contrario, se comprueba si el contador tiene un número de medidas aceptable, lo que significa que el usuario sigue moviéndose y lo que se hace es ir rutina a rutina de la lista, y se establece un radio de cierta holgura con centro en el destino de la rutina. De esta forma, somos capaces de considerar que el usuario está en camino siempre que permanezca dentro de ese radio, o si en este caso es una rutina nueva para almacenar.

Analizador

Esta tarea convierte lo que en un principio son muestras de latitud, longitud y fecha, en rutinas con una frecuencia, un lugar de partida y de llegada, y una convicción de fiabilidad de la misma. El proceso seguido para dicho análisis o aprendizaje, es el descrito por Davenport y Prusak, basado en la pirámide de conocimiento. Más

adelante encontramos una sección de procedimiento de análisis en la que se relatará con más detalle el algoritmo implementado que usa esta tarea.

Esta tarea solo se realizará una vez al día y en un momento preciso. Este momento es habitualmente cuando el dispositivo entra en modo de carga, lo cual significa que ha sido enchufado a la corriente, y por consiguiente que no va a tener repercusión en la batería del mismo.

Otra tarea importante que realiza este componente es el borrado de datos antiguos ya utilizados para minimizar el uso de la memoria. En esta fase está implementado el borrado de datos con más de dos meses de antigüedad.

Se ha implementado junto con la funcionalidad descrita la de recuperación ante un fallo. Para ello se realizan copias de seguridad de los datos automáticas o manuales, de este modo evitamos que algún tipo de error en el sistema a la hora de realizar la computación de los datos nos afecte.

7.2- Estructura de datos heredada

Podemos distinguir entre dos tipos diferentes de datos según la descripción de la estructura del sistema: el acumulador necesita almacenar las medidas que vaya recogiendo a lo largo del día; el analizador usar estos datos para crear lo que serían las rutinas, que también deben ser almacenadas.

Así la base de datos se compondría de tres tablas principalmente. Sin embargo, son necesarias otras tablas de apoyo, tanto para completar la estructura de una rutina, como de apoyo para el análisis. En la figura 9 se muestra el diagrama entidad/relación de la base de datos final con todas las tablas y sus relaciones.

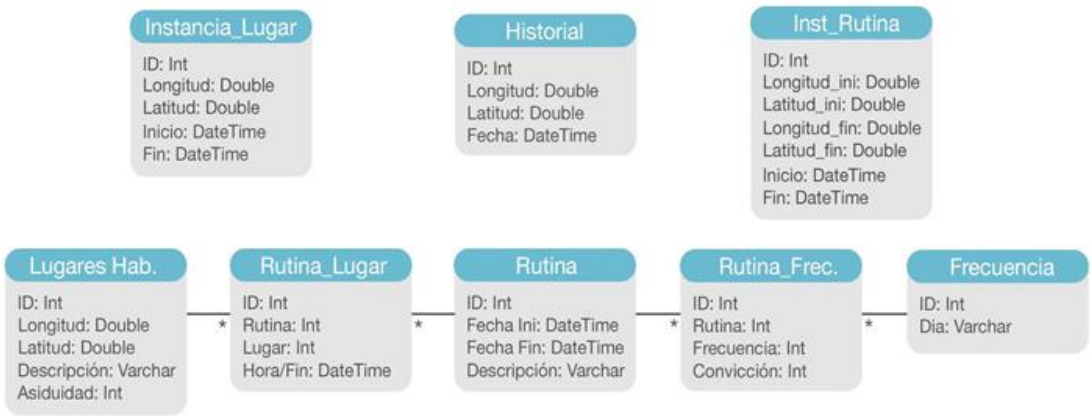

Figura 9: Diagrama entidad/relación base de datos

- **Historial**

Es donde el acumulador almacena todas las muestras que se van recogiendo. Esta tabla tiene los datos de ID, latitud, longitud y fecha.

- **Rutina**

En esta tabla se almacenan las rutinas y su grado de convicción y frecuencia. Contiene datos de:

-Lugares a los que el usuario acude durante la rutina.

-Hora de inicio de la rutina.

-Hora de fin de la rutina.

-Frecuencia semanal con la que se repite.

Los puntos a los que el usuario acude en la rutina se han modelado usando una relación muchos a muchos con una tabla llamada Lugares Habituales que contiene las coordenadas GPS de dichos puntos visitados por el usuario.

En cuanto a la columna frecuencia, también es una relación muchos a muchos con una tabla Día que contiene los nombres de los días de la semana.

- **Lugares Habituales**

Se trata de la tabla que se relaciona con las rutinas en la relación muchos a muchos de los lugares a los que irá acudiendo el usuario durante la misma. Representa los lugares en los que el usuario pasa más tiempo o acude con asiduidad. Esta tabla contendrá las columnas necesarias para identificar estos lugares.

-Coordenadas GPS.

-Descripción del lugar, o anotaciones que se quieran hacer sobre él.

-Asiduidad, que representará un número entero, representando el más alto, el lugar donde más ha ido el usuario, que suele ser su casa.

- **Frecuencia**

Es la tabla que está relacionada con Rutina, para representar la frecuencia semanal con la que se repite una rutina de desplazamiento del usuario. Solo contiene una fila por día de la semana de lunes a domingo. Para completar la información sobre una frecuencia semanal de una rutina, se añadió un campo de un entero para representar el grado de convicción que se tiene sobre la repetición del desplazamiento ese día. Este campo se encuentra en la relación muchos a muchos entre las dos tablas.

Por último, es de esperar que en análisis, antes de obtener las rutinas completas, son necesarios una serie de pasos intermedios en los que los datos van adquiriendo cada

vez una mayor complejidad. Estas tablas son, en orden ascendente de completitud, Instancia de Lugar e Instancia de Rutina.

- **Instancia de Lugar**

Es el primer paso en el análisis. En esta tabla se intentan identificar los lugares en los que el usuario ha permanecido cierto tiempo. Es el paso previo a convertirse en un lugar habitual. Contiene:

-Coordenadas GPS, del lugar en cuestión.

-Fecha y hora de comienzo, de la permanencia en ese lugar.

-Fecha y hora del fin, de la permanencia en ese lugar.

- **Instancia de Rutina**

Este es el último paso de los datos antes de convertirse en Rutinas. Estos datos ya tienen la misma forma que una rutina, a falta de la frecuencia.

-Coordenadas GPS, del lugar de partida.

-Coordenadas GPS, del lugar de llegada.

-Fecha y hora de salida, desde el lugar de partida.

-Fecha y hora de llegada, al destino.